The Solution Path of the Generalized Lasso

Ryan J. Tibshirani [*] Jonathan Taylor [†]

Abstract

We present a path algorithm for the generalized lasso problem. This problem penalizes the ℓ_1 norm of a matrix D times the coefficient vector, and has a wide range of applications, dictated by the choice of D. Our algorithm is based on solving the dual of the generalized lasso, which facilitates computation and conceptual understanding of the path. For $D = I$ (the usual lasso), we draw a connection between our approach and the well-known LARS algorithm. For an arbitrary D, we derive an unbiased estimate of the degrees of freedom of the generalized lasso fit. This estimate turns out to be quite intuitive in many applications.

Keywords: *lasso; path algorithm; Lagrange dual; LARS; degrees of freedom*

1 Introduction

Regularization with the ℓ_1 norm seems to be ubiquitous throughout many fields of mathematics and engineering. In statistics, the best-known example is the *lasso*, the application of an ℓ_1 penalty to linear regression [31, 7]. Let $y \in \mathbb{R}^n$ be a response vector and $X \in \mathbb{R}^{n\times p}$ be a matrix of predictors. If the response and the predictors have been centered, we can omit an intercept term from the model, and then the lasso problem is commonly written as

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

where $\lambda \geq 0$ is a tuning parameter. There are many fast algorithms for solving the lasso (1) at a single value of the parameter λ, or over a discrete set of parameter values. The *least angle regression* (LARS) algorithm, on the other hand, is unique in that it solves (1) for all $\lambda \in [0, \infty]$ [11] (see also the earlier *homotopy* method of [23], and the even earlier work of [3]). This is possible because the lasso solution is piecewise linear with respect to λ.

The LARS path algorithm may provide a computational advantage when the solution is desired at many values of the tuning parameter. For large problems, this is less likely to be the case because the number of knots (changes in slope) in the solution path tends to be very large, and this renders the path intractable. Computational efficiency aside, the LARS method fully characterizes the tradeoff between goodness-of-fit and sparsity in the lasso solution (this is controlled by λ), and hence yields interesting statistical insights into the problem. Most notably, the LARS paper established a result on the degrees of freedom of the lasso fit, which was further developed by [35].

The first of its kind, LARS inspired the development of path algorithms for various other optimization problems that appear in statistics [16, 25, 19, 20], and our case is no exception. In this paper, we derive a path algorithm for problems that use the ℓ_1 norm to enforce certain structural constraints—instead of pure sparsity—on the coefficients in a linear regression. These problems are nicely encapsulated by the formulation:

$$\underset{\beta\in\mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|D\beta\|_1, \tag{2}$$

[*]Dept. of Statistics, Stanford University. email: `ryantibs@stanford.edu`. Supported by NSF VIGRE fellowship.

[†]Dept. of Statistics, Stanford University. email: `jtaylo@stanford.edu`. Partially supported by NSF grants DMS-0852227 and DMS-0906801.

where $D \in \mathbb{R}^{m \times p}$ is a specified penalty matrix. We refer to problem (2) as the *generalized lasso*. Depending on the application, we choose D so that sparsity of $D\beta$ corresponds to some other desired behavior for β, typically one that is structural or geometric in nature. In fact, various choices of D in (2) give problems that are already well-known in the literature: the fused lasso, trend filtering, wavelet smoothing, and a method for outlier detection. We derive a simple path algorithm for the minimization (2) that applies to a general matrix D, hence this entire class of problems. Like the lasso, the generalized lasso solution is piecewise linear as a function of λ. We also prove a result on the degrees of freedom of the fit for a general D. It is worth noting that problem (2) has been considered by other authors, for example [28]. This last work establishes some asymptotic properties of the solution, and proposes a computational technique that relates to simulated annealing.

The paper is organized as follows. We begin in Section 2 by motivating the use of a penalty matrix D, offering several examples of problems that fit into this framework. Section 3 explains that some instances of the generalized lasso can be transformed into a regular lasso problem, but many instances cannot, which emphasizes the need for a new path approach. In Section 4 we derive the Lagrange dual of (2), which serves as the jumping point for our algorithm and all of the work that follows. For the sake of clarity, we build up the algorithm over the next 3 sections. Sections 5 and 6 consider the case $X = I$. In Section 5 we assume that D is the 1-dimensional fused lasso matrix, in which case our path algorithm takes an especially simple (and intuitive) form. In Section 6 we give the path algorithm for a general penalty matrix D, which requires adding only one step in the iterative loop. Section 7 extends the algorithm to the case of a general design matrix X. Provided that X has full column rank, we show that our path algorithm still applies, by rewriting the dual problem in a more familiar form. We also outline a path approach for the case when X has rank less than its number of columns. Practical considerations for the path's computation are given in Section 8.

In Section 9 we focus on the lasso case, $D = I$, and compare our method to LARS. Above, we described LARS as an algorithm for computing the solution path of (1). This actually refers to LARS in its "lasso" state, and although this is probably the best-known version of LARS, it is not the only one. In its original (unmodified) state, LARS does not necessarily optimize the lasso criterion, but instead performs a more "democratic" form of forward variable selection. It turns out that with an easy modification, our algorithm gives this selection procedure exactly. In Section 10 we derive an unbiased estimate of the degrees of freedom of the fit for a general matrix D. The proof is quite straightforward because it utilizes the dual fit, which is simply the projection onto a convex set. As we vary D, this result yields interpretable estimates of the degrees of freedom of the fused lasso, trend filtering, and more. Finally, Section 11 contains some discussion.

To save space (and improve readability), many of the technical details in the paper are deferred to a supplementary document, available online at `http://www-stat.stanford.edu/~ryantibs/`.

2 Applications

There are a wide variety of interesting applications of problem (2). What we present below is not meant to be an exhaustive list, but rather a set of illustrative examples that motivated our work on this problem in the first place. This section is split into two main parts: the case when $X = I$ (often called the "signal approximation" case), and the case when X is a general design matrix.

2.1 The signal approximation case, $X = I$

When $X = I$, the solution of the lasso problem (1) is given by soft-thresholding the coordinates of y. Therefore one might think that an equally simple formula exists for the generalized lasso solution when the design matrix is the identity—but this is not true. Taking $X = I$ in the generalized lasso (2) gives an interesting and highly nontrivial class of problems. In this setup, we observe data $y \in \mathbb{R}^n$

which is a noisy realization of an underlying signal, and the rows of $D \in \mathbb{R}^{m \times n}$ reflect some believed structure or geometry in the signal. The solution of problem (2) fits adaptively to the data while exhibiting some of these structural properties. We begin by looking at piecewise constant signals, and then address more complex features.

2.1.1 The fused lasso

Suppose that y follows a 1-dimensional structure, that is, the coordinates of y correspond to successive positions on a straight line. If D is the $(n-1) \times n$ matrix

$$D_{\mathrm{1d}} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}, \tag{3}$$

then problem (2) penalizes the absolute differences in adjacent coordinates of β, and is known as the *1d fused lasso* [32]. This gives a piecewise constant fit, and is used in settings where coordinates in the true model are closely related to their neighbors. A common application area is comparative genomic hybridization (CGH) data: here y measures the number of copies of each gene ordered linearly along the genome (actually y is the log ratio of the number of copies relative to a normal sample), and we believe for biological reasons that nearby genes will exhibit a similar copy number. Identifying abnormalities in copy number has become a valuable means of understanding the development of many human cancers. See Figure 1 for an example of the 1d fused lasso applied to some CGH data on glioblastoma multiformes, a particular type of malignant brain tumor, taken from [5].

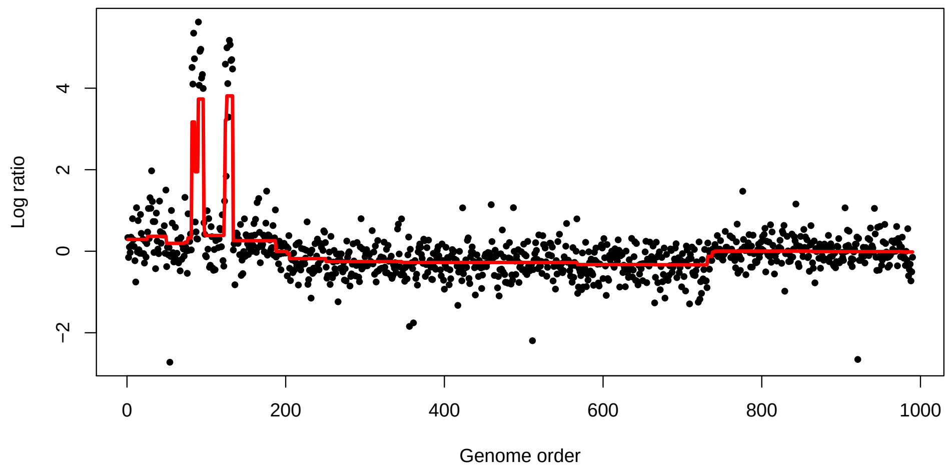


Figure 1: *The 1d fused lasso applied to some glioblastoma multiforme data. The red line represents the inferred copy number from the 1d fused lasso solution (for $\lambda = 3$).*

A natural extension of this idea penalizes the differences between neighboring pixels in an image. Suppose that y represents a noisy image that has been unraveled into a vector, and each row of D again has a 1 and -1, but this time arranged to give both the horizontal and vertical differences between pixels. Then problem (2) is called the *2d fused lasso* [32], and is used to denoise images that we believe should obey a piecewise constant structure. This technique is a special type of *total variation denoising*, a well-studied problem that carries a vast literature spanning the fields of statistics, computer science, electrical engineering, and others (for example, see [26]). Figure 2 shows the 2d fused lasso applied to a toy example.

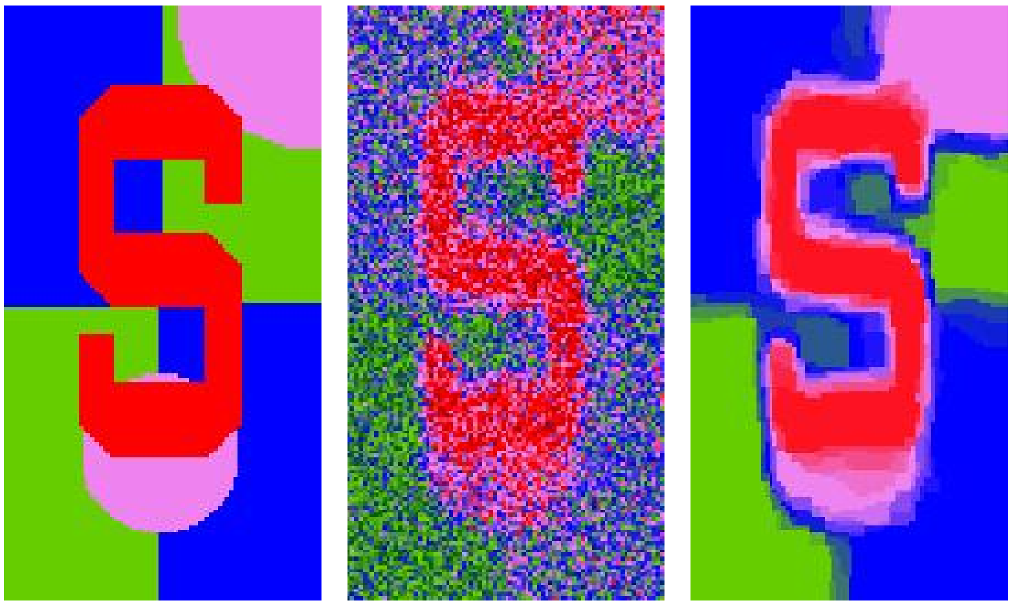

(a) Original (b) Noisy (c) Denoised

Figure 2: *An example of the 2d fused lasso for image denoising. We started with a toy signal, shown in (a). The colors green, blue, purple, red in the image correspond to the numeric levels* $1, 2, 3, 4$*, respectively. We then added noise, shown in (b), interpolating between colors to display the intermediate values. This is used as the data* y *in the 2d fused lasso problem. The solution (for* $\lambda = 1$*) is shown in (c), and it is a fairly accurate reconstruction. The fused lasso is effective here because the original image is piecewise constant.*

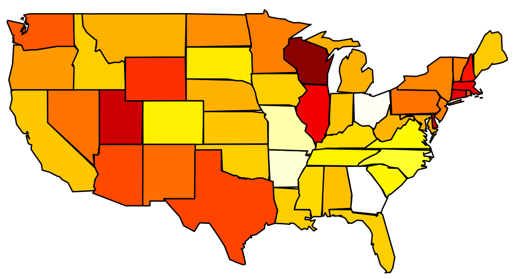
We can further extend this idea by defining adjacency according to an arbitrary graph structure, with n nodes and m edges. Now the coordinates of $y \in \mathbb{R}^n$ correspond to nodes in the graph, and we penalize the difference between each pair of nodes joined by an edge. Hence D is $m \times n$, with each row having a -1 and 1 in the appropriate spots, corresponding to an edge in the graph. In this case, we simply call problem (2) the *fused lasso*. Note that both the 1d and 2d fused lasso problems are special cases of this, with the underlying graph a chain and a 2d grid, respectively. But the fused lasso is a very general problem, as it can be applied to any graph structure that exhibits a piecewise constant signal across adjacent nodes. See Figure 3 for application in which the underlying graph has US states as nodes, with two states joined by an edge if they share a border. This graph has 48 nodes (we only include the mainland US states) and 105 edges.

The observant reader may notice a discrepancy between the usual fused lasso definition and ours, as the fused lasso penalty typically includes an additional term $\|\beta\|_1$, the ℓ_1 norm of the coefficients themselves. We refer to this as the *sparse fused lasso*, and to represent this penalty we just append the $n \times n$ identity matrix to the rows of D. Actually, this carries over to all of the applications yet to be discussed—if we desire pure sparsity in addition to the structural behavior that is being encouraged by D, we append the identity matrix to the rows of D.

2.1.2 Linear and polynomial trend filtering

Suppose again that y follows a 1-dimensional structure, but now D is the $(n-2) \times n$ matrix

$$D_{\mathrm{tf},1} = \begin{bmatrix} -1 & 2 & -1 & \ldots & 0 & 0 & 0 \\ 0 & -1 & 2 & \ldots & 0 & 0 & 0 \\ \ldots & & & & & & \\ 0 & 0 & 0 & \ldots & -1 & 2 & -1 \end{bmatrix}.$$


(a) Data

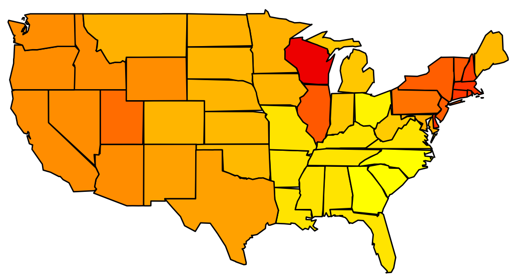
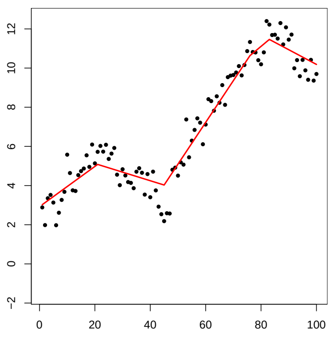

(b) Fused Lasso Solution

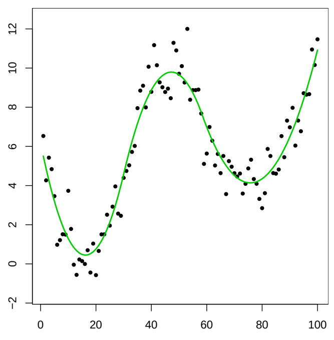
Figure 3: *An example of the fused lasso on an irregular graph. The data y are the log proportion of H1N1 flu cases for each (mainland) US state in the year 2009, shown in (a). This was taken from [6]. The color map uses white to reflect the lowest measured log proportion, and dark red to reflect the highest, with yellow, orange, and red in between. We can think of the data as noisy measurements of the true log probabilities of infection in each state, which likely exhibits some geographic trend. Therefore, we solve the fused lasso problem on a custom underlying graph, where we connect two states by an edge if they share a border. Shown in (b) is the solution (for $\lambda = 0.25$). Here, groups of states are assigned the same color or "fused" on the west coast, in the mid west, in the south east, and in the north east. The colors suggest that, among these regions, you are most likely to get H1N1 flu if you live in the north east, then the west coast, then the midwest, and then the south east. But there certainly are states that don't get fused into these regions, like Wisconsin and Illinois, where the infection rates are exceptionally high.*

Then problem (2) is equivalent to *linear trend filtering* (also called *ℓ_1 trend filtering*) [21]. Just as the 1d fused lasso penalizes the discrete first derivative, this technique penalizes the discrete second derivative, and so it gives a piecewise linear fit. This has many applications, namely, any settings in which the underlying trend is believed to be linear with (unknown) changepoints. Moreover, by recursively defining

$$D_{\mathrm{tf},k} = D_{\mathrm{1d}} \cdot D_{\mathrm{tf},k-1} \quad \text{for} \quad k = 2, 3, \ldots,$$

(here D_{1d} is the $(n-k-1) \times (n-k)$ version of (3)) we can fit a piecewise polynomial of any order k, further extending the realm of applications. We call this *polynomial trend filtering of order k*. Figure 4 shows examples of linear, quadratic, and cubic fits.


(a) Linear


(b) Quadratic

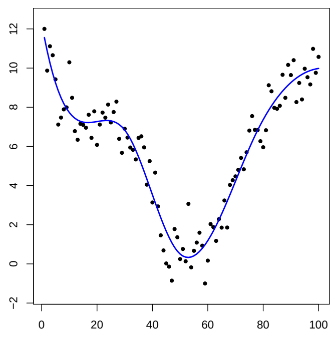

(c) Cubic

Figure 4: *Solutions of* (2) *for three problems, with D equal to (a) $D_{\mathrm{tf},1}$, (b) $D_{\mathrm{tf},2}$, and (c) $D_{\mathrm{tf},3}$. These are piecewise linear, quadratic, and cubic, respectively. (For each problem we chose a different value of the regularization parameter λ.)*

The polynomial trend filtering fits (especially for $k = 3$) are similar to those that one could obtain using regression splines and smoothing splines. However, the knots (changes in kth derivative) in the trend filtering fits are selected adaptively based on the data, jointly with the inter-knot polynomial estimation. This phenomenon of simultaneous selection and estimation—analogous to that concerning the nonzero coefficients in the lasso fit, and the jumps in the piecewise constant fused lasso fit—does not occur in regression and smoothing splines. Regression splines operate on a fixed set of knots, and there is a substantial literature on knot placement for this problem (see Chapter 9.3 of [17], for example). Smoothing splines place a knot at each data point, and implement smoothness via a generalized ridge regression on the coefficients in a natural spline basis. As a result (of this ℓ_2 shrinkage), they cannot represent both global smoothness and local wiggliness in a signal. On the other hand, trend filtering has the potential to represent both such features, a property called "time and frequency localization" in the signal processing field, though this idea has been largely unexplored. The classic example of a procedure that allows time and frequency localization is wavelet smoothing, discussed next.

2.1.3 Wavelet smoothing

This is a quite a popular method in signal processing and compression. The main idea is to model the data as a sparse linear combination of wavelet functions. Perhaps the most common formulation for wavelet smoothing is *SURE shrinkage* [9], which solves the lasso optimization problem

$$\underset{\theta \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|y - W\theta\|_2^2 + \lambda\|\theta\|_1, \tag{4}$$

where $W \in \mathbb{R}^{n\times n}$ has an orthogonal wavelet basis along its columns. By orthogonality, we can change variables to $\beta = W\theta$ and then (4) becomes a generalized lasso problem with $D = W^T$.

In many applications it is desirable to use an overcomplete wavelet set, so that $W \in \mathbb{R}^{n\times m}$ with $n < m$. Now problem (4) and the generalized lasso (2) with $D = W^T$ (and $X = I$) are no longer equivalent, and in fact give quite different answers. In signal processing, the former is called the *synthesis* approach, and the latter the *analysis* approach, to wavelet smoothing. Though attention has traditionally been centered around synthesis, a recent paper by [12] suggests that synthesis may be too sensitive, and shows that it can be outperformed by its analysis counterpart.

2.2 A general design matrix X

For any of the fused lasso, trend filtering, or wavelet smoothing penalties discussed above, the addition of a general matrix X of covariates significantly extends the domain of applications. For a fused lasso example, suppose that each row of X represents a $k_1 \times k_2 \times k_3$ MRI image of a patient's brain, unraveled into a vector (so that $p = k_1 \cdot k_3 \cdot k_3$). Suppose that y contains some continuous outcome on the patients, and we model these as a linear function of the MRIs, $\mathrm{E}(y_i|X_i) = \beta^T X_i$. Now β also has the structure of a $k_1 \times k_2 \times k_3$ image, and by choosing the matrix D to give the sparse 3d fused lasso penalty (that is, the fused lasso on a 3d grid with an additional ℓ_1 penalty of the coefficients), the solution of (2) attempts to explain the outcome with a small number of contiguous regions in the brain.

As another example, the inclusion of a design matrix X in the trend filtering setup provides an alternative way of fitting *varying-coefficient models* [18, 8]. We consider a data set from [18], which examines $n = 88$ observations on the exhaust from an engine fueled by ethanol. The response y is the concentration of nitrogen dioxide, and the two predictors are a measure of the fuel-air ratio E, and the compression ratio of the engine C. Studying the interactions between E and C leads the authors of [18] to consider the model

$$\mathrm{E}(y_i|E_i, C_i) = \beta_0(E_i) + \beta_1(E_i) \cdot C_i. \tag{5}$$

This is a linear model with a different intercept and slope for each E_i, subject to the (implicit) constraint that the intercept and slope should vary smoothly along the E_i's. We can fit this using

(2), in the following way: first we discretize the continuous observations $E_1, \dots E_n$ so that they fall into, say, 25 bins. Our design matrix X is 88×50, with the first 25 columns modeling the intercept β_0 and the last 25 modeling the slope β_1. The ith row of X is

$$X_{ij} = \begin{cases} 1 & \text{if } E_i \text{ lies in the } j\text{th bin} \\ C_i & \text{if } E_i \text{ lies in the } (j+25)\text{th bin} \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we choose

$$D = \begin{bmatrix} D_{\text{tf},3} & 0 \\ 0 & D_{\text{tf},3} \end{bmatrix},$$

where $D_{\text{tf},3}$ is the cubic trend filtering matrix (the choice $D_{\text{tf},3}$ is not crucial and of course can be replaced by a higher or lower order trend filtering matrix.) The matrix D is structured in this way so that we penalize the smoothness of the first 25 and last 25 components of $\beta = (\beta_0, \beta_1)^T$ individually. With X and D as described, solving the optimization problem (2) gives the coefficients shown in Figure 5, which appear quite similar to those plotted in [18].

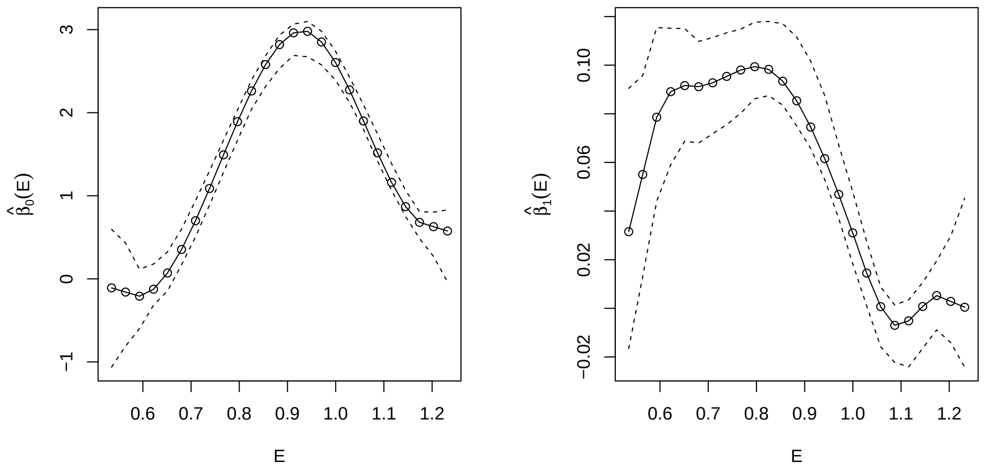


Figure 5: *The intercept and slope of the varying-coefficient model* (5) *for the engine data of [18], fit using* (2) *with a cubic trend filtering penalty matrix (and* $\lambda = 3$*). The dashed lines show 85% bootstrap confidence intervals from 500 bootstrap samples.*

We conclude this section with a generalized lasso application of [29], in which the penalty is not structurally-based, unlike the examples discussed previously. Suppose that we observe $y_1, \dots y_n$, and we believe the majority of these points follow a linear model $\mathrm{E}(y_i|X_i) = \beta^T X_i$ for some covariates $X_i = (X_{i1}, \dots X_{ip})^T$, except that a small number of the y_i are outliers and do not come from this model. To determine which points are outliers, one might consider the problem

$$\underset{z \in \mathbb{R}^n,\, \beta \in \mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\|z - X\beta\|_2^2 \ \text{ subject to } \|z - y\|_0 \le k \tag{6}$$

for a fixed integer k. Here $\|x\|_0 = \sum_i 1(x_i \neq 0)$. Thus by setting $k = 3$, for example, the solution $\hat{z}$ of (6) would indicate which 3 points should be considered outliers, in that $\hat{z}_i \neq y_i$ for exactly 3 coordinates. A natural convex relaxation of problem (6) is

$$\underset{z \in \mathbb{R}^n,\, \beta \in \mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\|z - X\beta\|_2^2 + \lambda\|z - y\|_1, \tag{7}$$

where we have also transformed the problem from bound form to Lagrangian form. Letting $\alpha = y - z$, this can be rewritten as

$$\underset{\alpha\in\mathbb{R}^n,\,\beta\in\mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\|y - \alpha - X\beta\|_2^2 + \lambda\|\alpha\|_1, \tag{8}$$

which fits into the form of problem (2), with design matrix $\widetilde{X} = [\,I \;\; X\,]$, coefficient vector $\tilde{\beta} = (\alpha, \beta)^T$, and penalty matrix $D = [\,I \;\; 0\,]$. Figure 6 shows a simple example with $p = 1$.

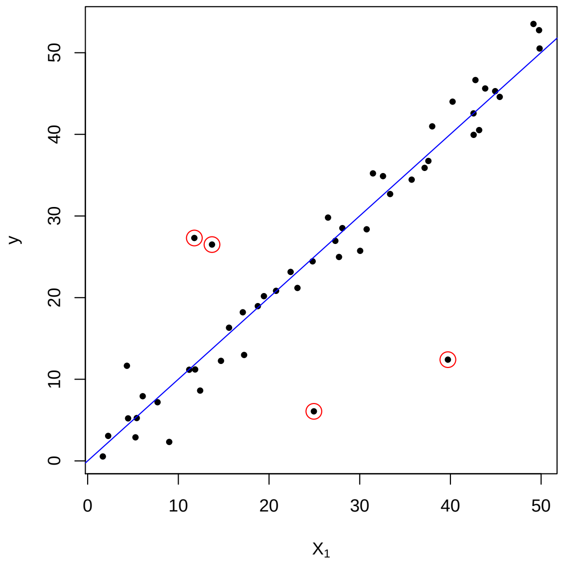


Figure 6: *A simple example of using problem* (2) *to perform outlier detection. Written in the form* (8)*, the blue line denotes the fitted slope $\hat{\beta}$, while the red circles indicate the outliers, as determined by the coordinates of $\hat{\alpha}$ that are nonzero (for $\lambda = 8$).*

After reading the examples in this section, a natural question is: when can a generalized lasso problem (2) be transformed into a regular lasso problem (1)? (Recall, for example, that this is possible for an orthogonal D, as we discussed in the wavelet smoothing example.) We discuss this in the next section.

3 When does a generalized lasso problem reduce to a lasso problem?

If D is $p \times p$ and invertible, we can transform variables in problem (2) by $\theta = D\beta$, yielding the lasso problem

$$\underset{\theta\in\mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\|y - XD^{-1}\theta\|_2^2 + \lambda\|\theta\|_1. \tag{9}$$

More generally, if D is $m \times p$ and $\text{rank}(D) = m$ (note that this necessarily means $m \le p$), then we can still transform variables and get a lasso problem. First we construct a $p \times p$ matrix $\widetilde{D} = \begin{bmatrix} D \\ A \end{bmatrix}$ with $\text{rank}(\widetilde{D}) = p$, by finding a $(p-m) \times p$ matrix A whose rows are orthogonal to those in D. Then we change variables to $\theta = (\theta_1, \theta_2)^T = \widetilde{D}\beta$, so that the generalized lasso (2) becomes

$$\underset{\theta\in\mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\|y - X\widetilde{D}^{-1}\theta\|_2^2 + \lambda\|\theta_1\|_1. \tag{10}$$

This is almost a regular lasso, except that the ℓ_1 penalty only covers part of the coefficient vector. First write $X\widetilde{D}^{-1}\theta = X_1\theta_1 + X_2\theta_2$; then, it is clear that at the solution the second block of the coefficients is given by a linear regression:

$$\hat{\theta}_2 = (X_2^T X_2)^{-1} X_2^T (y - X_1\hat{\theta}_1).$$

Therefore we can rewrite problem (10) as

$$\underset{\theta_1 \in \mathbb{R}^m}{\text{minimize}} \ \frac{1}{2}\|(I-P)y - (I-P)X_1\theta_1\|_2^2 + \lambda\|\theta_1\|_1, \tag{11}$$

where $P = X_2^T(X_2^T X_2)^{-1}X_2^T$, the projection onto the column space of X_2. The LARS algorithm provides the solution path of such a lasso problem (11), from which we can back-transform to get the generalized lasso solution: $\hat{\beta} = \widetilde{D}^{-1}\hat{\theta}$.

However, if D is $m \times p$ and $\text{rank}(D) < m$, then such a transformation is not possible, and LARS cannot be used to find the solution path of the generalized lasso problem (2). Further, in this case, the authors of [12] establish what they call an "unbridgeable" gap between problems (1) and (2), based on the geometric properties of their solutions.

While several of the examples from Section 2 satisfy $\text{rank}(D) = m$, and hence admit a lasso transformation, a good number also fall into the case $\text{rank}(D) < m$, and suggest the need for a novel path algorithm. These are summarized in Table 1. Therefore, in the next section, we derive the Lagrange dual of problem (2), which leads to a nice algorithm to compute the solution path of (2) for an arbitrary penalty matrix D.

$\text{rank}(D) = m$	$\text{rank}(D) < m$
• The 1d fused lasso	• The fused lasso on any graph that has more edges than nodes (for example, the 2d fused lasso)
• Polynomial trend filtering of any order	• The sparse fused lasso on any graph
• Wavelet smoothing with an orthogonal wavelet basis	• Wavelet smoothing with an overcomplete wavelet set
• Outlier detection	

Table 1: *Examples from Section 2 that fall into the cases* $\text{rank}(D) = m$ *and* $\text{rank}(D) < m$.

4 The Lagrange dual problem

Roughly speaking, the generalized lasso problem (2) is difficult to analyze directly because the nondifferentiable ℓ_1 penalty is composed with a linear transformation of β. We hence turn to the corresponding Lagrange dual problem where the story is conceptually clearer. First we consider the generalized lasso in the signal approximation case, $X = I$:

$$\underset{\beta \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|y - \beta\|_2^2 + \lambda\|D\beta\|_1. \tag{12}$$

Following an argument of [21], we rewrite this problem as

$$\underset{\beta \in \mathbb{R}^n,\, z \in \mathbb{R}^m}{\text{minimize}} \ \frac{1}{2}\|y - \beta\|_2^2 + \lambda\|z\|_1 \ \text{ subject to } \ z = D\beta.$$

The Lagrangian is hence

$$\mathcal{L}(\beta, z, u) = \frac{1}{2}\|y - \beta\|_2^2 + \lambda\|z\|_1 + u^T(D\beta - z),$$

and to derive the dual problem, we minimize this over β, z. The terms involving β are just a quadratic, and up to some constants (not depending on u)

$$\min_\beta \left(\frac{1}{2}\|y - \beta\|_2^2 + u^T D\beta\right) = -\frac{1}{2}\|y - D^T u\|_2^2,$$

while

$$\min_z \left(\lambda\|z\|_1 - u^T z\right) = \begin{cases} 0 & \text{if } \|u\|_\infty \le \lambda \\ -\infty & \text{otherwise.} \end{cases}$$

Therefore the dual problem of (12) is

$$\underset{u \in \mathbb{R}^m}{\text{minimize}} \;\; \frac{1}{2}\|y - D^T u\|_2^2 \;\; \text{subject to} \;\; \|u\|_\infty \le \lambda. \tag{13}$$

Immediately we can see that (13) has a "nice" constraint set, $\{u : \|u\|_\infty \le \lambda\}$, which is simply a box, free of any linear transformation. It is also important to note the difference in dimension: the dual problem has a variable $u \in \mathbb{R}^m$, whereas the original problem (12), called the primal problem, has a variable $\beta \in \mathbb{R}^n$.

When $\text{rank}(D) < m$, the dual problem is not strictly convex, and so it can have many solutions. On the other hand, the primal problem is always strictly convex and always has a unique solution. The primal problem is also strictly feasible (it has no constraints), and so strong duality holds (see Section 5.2 of [4]). Let us denote the solutions of the primal problem (12) and dual problem (13) by $\hat{\beta}_\lambda$ and $\hat{u}_\lambda$, respectively, to emphasize their dependence on λ. By taking the gradient of the Lagrangian $\mathcal{L}(\beta, z, u)$ with respect to β and setting this equal to zero, we get the primal-dual relationship

$$\hat{\beta}_\lambda = y - D^T \hat{u}_\lambda. \tag{14}$$

Taking the gradient of $\mathcal{L}(\beta, z, u)$ with respect to z and setting this equal to zero, we find that each coordinate $i = 1, \ldots m$ of the dual solution satisfies

$$\hat{u}_{\lambda,i} \in \begin{cases} \{+\lambda\} & \text{if } (D\hat{\beta}_\lambda)_i > 0 \\ \{-\lambda\} & \text{if } (D\hat{\beta}_\lambda)_i < 0 \\ [-\lambda, \lambda] & \text{if } (D\hat{\beta}_\lambda)_i = 0. \end{cases} \tag{15}$$

This last equation tells us that the dual coordinates that are equal to λ in absolute value,

$$\mathcal{B} = \{i : |\hat{u}_{\lambda,i}| = \lambda\}, \tag{16}$$

are the coordinates of $D\hat{\beta}_\lambda$ that are "allowed" to be nonzero. But this does necessarily mean that $(D\hat{\beta}_\lambda)_i \ne 0$ for all $i \in \mathcal{B}$.

For a general design matrix X, we can apply a similar argument to derive the dual of (2):

$$\begin{aligned} \underset{u \in \mathbb{R}^m}{\text{minimize}} \;\; & \frac{1}{2}(X^T y - D^T u)^T (X^T X)^+ (X^T y - D^T u) \\ \text{subject to} \;\; & \|u\|_\infty \le \lambda, \; D^T u \in \text{row}(X), \end{aligned} \tag{17}$$

This looks complicated, certainly in comparison to problem (13). However, the inequality constraint on u is still a simple (un-transformed) box. Moreover, we can make (17) look like (13) by changing the response y and penalty matrix D. This will be discussed later in Section 7.

In the next two sections, Sections 5 and 6, we restrict our attention to the case $X = I$ and derive an algorithm to find a solution path of the dual (13). This gives the desired primal solution path, using the relationship (14). Since our focus is on solving the dual problem, we write simply "solution" or "solution path" to refer to the dual versions. Though we will eventually consider an arbitrary matrix D in Section 6, we begin by studying the 1d fused lasso in Section 5. This case is especially simple, and we use it to build the framework for the path algorithm in the general D case.

5 The 1d fused lasso

In this setting we have $D = D_{\text{1d}}$, the $(n-1) \times n$ matrix given in (3). Now the dual problem (13) is strictly convex (since D_{1d} has rank equal to its number of rows), and therefore it has a unique solution. In order to efficiently compute the solution path, we use a lemma that allows us, at different stages, to reduce the dimension of the problem by one.

5.1 The boundary lemma

Consider the constraint set $\{u : \|u\|_\infty \leq \lambda\} \subseteq \mathbb{R}^{n-1}$: this is a box centered around the origin with side length 2λ. We say that coordinate i of u is "on the boundary" (of this box) if $|u_i| = \lambda$. For the 1d fused lasso, it turns out that coordinates of the solution that are on the boundary will remain on the boundary indefinitely as λ decreases. This idea can be stated more precisely as follows:

Lemma 1 (**The boundary lemma**). *Suppose that $D = D_{\text{1d}}$, the 1d fused lasso matrix in* (3). *For any coordinate i, the solution $\hat{u}_\lambda$ of* (13) *satisfies*

$$\hat{u}_{\lambda_0,i} = \lambda_0 \;\;\Rightarrow\;\; \hat{u}_{\lambda,i} = \lambda \;\; \textit{for all} \;\; 0 \leq \lambda \leq \lambda_0,$$

and

$$\hat{u}_{\lambda_0,i} = -\lambda_0 \;\;\Rightarrow\;\; \hat{u}_{\lambda,i} = -\lambda \;\; \textit{for all} \;\; 0 \leq \lambda \leq \lambda_0.$$

The proof is given in the online supplement. It is interesting to note a connection between the boundary lemma and a lemma of [14], which states that

$$\hat{\beta}_{\lambda_0,i} = \hat{\beta}_{\lambda_0,i+1} \;\;\Rightarrow\;\; \hat{\beta}_{\lambda,i} = \hat{\beta}_{\lambda,i+1} \;\; \text{for all} \;\; \lambda \geq \lambda_0 \tag{18}$$

for this same problem. In other words, this lemma says that no two equal primal coordinates can become unequal with increasing λ. In general $|\hat{u}_{\lambda,i}| = \lambda$ is not equivalent to $(D\hat{\beta}_\lambda)_i \neq 0$, but these two statements are equivalent for the 1d fused lasso problem (see the primal-dual correspondence in Section 5.3), and therefore the boundary lemma is equivalent to (18).

5.2 Path algorithm

This section is intended to explain the path algorithm from a conceptual point of view, and no rigorous arguments for its correctness are made here. We defer these until Section 6.1, when we revisit the problem in the context of a general matrix D.

The boundary lemma describes the behavior of the solution as λ decreases, and therefore it is natural to construct the solution path by moving the parameter from $\lambda = \infty$ to $\lambda = 0$. As will be made apparent from the details of the algorithm, the solution path is a piecewise linear function of λ, with a change in slope occurring whenever one of its coordinate paths hits the boundary. The key observation is that, by the boundary lemma, if a coordinate hits the boundary it will stay on the boundary for the rest of the path down to $\lambda = 0$. Hence when it hits the boundary we can essentially eliminate this coordinate from consideration (since we know its value at each smaller λ), recompute the slopes of the other coordinate paths, and move until another coordinate hits the boundary.

As we construct the path, we maintain two lists: $\mathcal{B} = \mathcal{B}(\lambda)$, which contains the coordinates that are currently on the boundary; and $s = s(\lambda)$, which contains their signs. For example, if we have $\mathcal{B}(\lambda) = (5, 2)$ and $s(\lambda) = (-1, 1)$, then this means that $\hat{u}_{\lambda,5} = -\lambda$ and $\hat{u}_{\lambda,2} = \lambda$. We call the coordinates in $\mathcal{B}$ the "boundary coordinates", and the rest the "interior coordinates". Now we can describe the algorithm:

Algorithm 1 (Dual path algorithm for the 1d fused lasso).

- *Start with $\lambda_0 = \infty$, $\mathcal{B} = \emptyset$, and $s = \emptyset$.*
- *For $k = 0, \ldots n-2$:*
  1. *Compute the solution at λ_k by least squares, as in* (20)*.*
  2. *Continuing in a linear direction from the solution, compute λ_{k+1}, when an interior coordinate will next hit the boundary, as in* (21) *and* (22)*.*
  3. *Add this coordinate to $\mathcal{B}$ and its sign to s.*

The algorithm's details appear slightly more complicated, but this is only because of notation. If $\mathcal{B} = (i_1, \ldots i_k)$, then we define for a matrix A and a vector x

$$A_{\mathcal{B}} = \begin{bmatrix} A_{i_1} \\ \vdots \\ A_{i_k} \end{bmatrix} \quad \text{and} \quad x_{\mathcal{B}} = (x_{i_1}, \ldots x_{i_k})^T,$$

where A_i is the ith row of A. In words: $A_{\mathcal{B}}$ gives the rows of A that are in $\mathcal{B}$, and $x_{\mathcal{B}}$ gives the coordinates of x in $\mathcal{B}$. We use the subscript $-\mathcal{B}$, as in $A_{-\mathcal{B}}$ or $x_{-\mathcal{B}}$, to index over the rows or coordinates except those in $\mathcal{B}$. Note that $\mathcal{B}$ as defined above (in the paragraph preceding the algorithm) is consistent with our previous definition (16), except that here we treat $\mathcal{B}$ as an ordered list instead of a set (its ordering only needs to be consistent with that of s). Also, we treat s as a vector when convenient.

When $\lambda = \infty$, the problem is unconstrained, and so clearly $\mathcal{B} = \emptyset$ and $s = \emptyset$. But more generally, suppose that we are at the kth iteration, with boundary set $\mathcal{B} = \mathcal{B}(\lambda_k)$ and signs $s = s(\lambda_k)$. By the boundary lemma, the solution satisfies

$$\hat{u}_{\lambda,\mathcal{B}} = \lambda s \quad \text{for all} \quad \lambda \in [0, \lambda_k].$$

Therefore, for $\lambda \leq \lambda_k$, we can reduce the optimization problem (13) to

$$\underset{u_{-\mathcal{B}}}{\text{minimize}} \ \frac{1}{2}\|y - \lambda(D_{\mathcal{B}})^T s - (D_{-\mathcal{B}})^T u_{-\mathcal{B}}\|_2^2 \quad \text{subject to} \quad \|u_{-\mathcal{B}}\|_\infty \leq \lambda, \tag{19}$$

which involves solving for just the interior coordinates. By construction, $\hat{u}_{\lambda_k,-\mathcal{B}}$ lies strictly between $-\lambda_k$ and λ_k in every coordinate. Therefore it is found by simply minimizing the objective function in (19), which gives the least squares estimate

$$\hat{u}_{\lambda_k,-\mathcal{B}} = \big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^{-1} D_{-\mathcal{B}}\big(y - \lambda_k(D_{\mathcal{B}})^T s\big). \tag{20}$$

Now let $a - \lambda_k b$ denote the right-hand side above. For $\lambda \leq \lambda_k$, the interior solution will continue to be $\hat{u}_{\lambda,-\mathcal{B}} = a - \lambda b$ until one of its coordinates hits the boundary. This critical value is determined by solving, for each i, the equation $a_i - \lambda b_i = \pm\lambda$; a simple calculation shows that the solution is

$$t_i = \frac{a_i}{b_i \pm 1} = \frac{\Big[\big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^{-1} D_{-\mathcal{B}} y\Big]_i}{\Big[\big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^{-1} D_{-\mathcal{B}}(D_{\mathcal{B}})^T s\Big]_i \pm 1} \tag{21}$$

(only one of $+1$ or -1 will yield a value $t_i \in [0, \lambda_k]$), which we call the "hitting time" of coordinate i. We take λ_{k+1} to be maximum of these hitting times

$$\lambda_{k+1} = \max_i \, t_i. \tag{22}$$

Then we compute

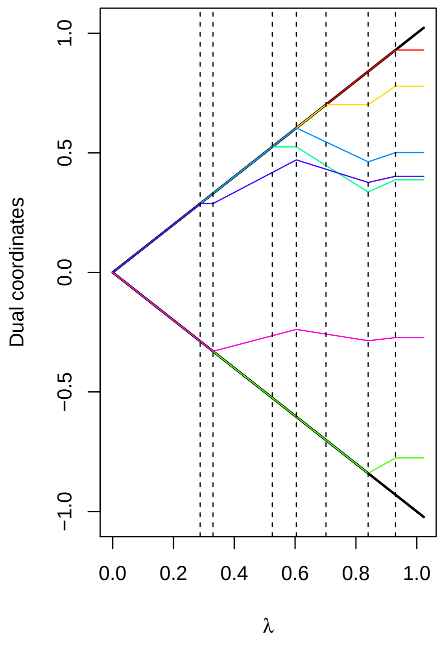
$$i_{k+1} = \underset{i}{\operatorname{argmax}} \; t_i \quad \text{and} \quad s_{k+1} = \operatorname{sign}\big(\hat{u}_{\lambda_{k+1}, i_{k+1}}\big),$$

and append i_{k+1} and s_{k+1} to $\mathcal{B}$ and s, respectively.

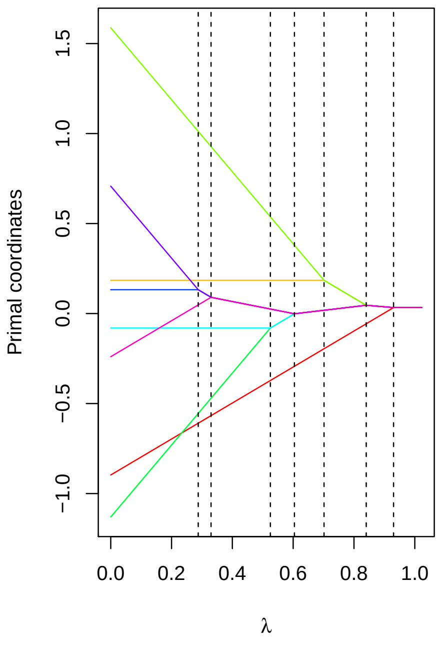
5.3 Properties of the solution path

Here we study some of the path's basic properties. Again we defer any rigorous arguments until Section 6.2, when we consider a general penalty matrix D. Instead we demonstrate them by way of a simple example.

Consider Figure 7(a), which shows the coordinate paths $\hat{u}_{\lambda,i}$ for an example with $n = 8$. Recall that it is natural to interpret the paths from right to left ($\lambda = \infty$ to $\lambda = 0$). Initially all of the slopes are zero, because when $\lambda = \infty$ the solution is just the least squares estimate $(DD^T)^{-1}Dy$, which has no dependence on λ. When a coordinate path first hits the boundary (the topmost path, drawn in red) the slopes of the other paths change, and they don't change again until another coordinate hits the boundary (the bottommost path, drawn in green), and so on, until all coordinates are on the boundary.



(a) Dual



(b) Primal

Figure 7: *Dual and primal coordinate paths for a small problem with $n = 8$.*

The picture suggests that the path $\hat{u}_\lambda$ is continuous and piecewise linear with respect to λ, with changes in slope or "kinks" at the values $\lambda_1, \ldots \lambda_{n-1}$ visited by the algorithm. (Piecewise linearity is obvious from the algorithm's construction of the path, but continuity is not.) This is also true in the general D case, although the solution path can have more than m kinks for an $m \times n$ matrix D.

On the other hand, Figure 7(b) shows the corresponding primal coordinate paths

$$\hat{\beta}_{\lambda,i} = (y - D^T \hat{u}_\lambda)_i.$$

As $\hat{u}_\lambda$ is a continuous piecewise linear function of λ, so is $\hat{\beta}_\lambda$, again with kinks at $\lambda_1, \ldots \lambda_{n-1}$. In contrast to the dual versions, it is natural to interpret the primal coordinate paths from left to right, because in this direction the coordinate paths become adjoined, or "fused", at some values of λ. The primal picture suggests that these fusion values are the same as the kinks $\lambda_1, \ldots \lambda_{n-1}$, that is:

- *Primal-dual correspondence for the 1d fused lasso.* The values of λ at which two primal coordinates fuse are exactly the values of λ at which a dual coordinate hits the boundary.

A similar property holds for the fused lasso on an arbitrary graph, though the primal-dual correspondence is a little more complicated for this case.

Note that as λ decreases in Figure 7(a), no dual coordinate paths leave the boundary. This is prescribed by the boundary lemma. As λ increases in Figure 7(b), no primal two coordinates split apart, or "un-fuse". This is prescribed by a lemma of [14] that we paraphrased in (18), and the two lemmas are equivalent.

6 A general penalty matrix D

Now we consider (13) for general $m \times n$ matrix D. The first question that comes to mind is: does the boundary lemma still hold? If DD^T is diagonally dominant, that is

$$(DD^T)_{ii} \geq \sum_{j \neq i} |(DD^T)_{ij}| \quad \text{for } i = 1, \ldots m, \tag{23}$$

then indeed the boundary lemma is still true. (See the online supplement.) Therefore the path algorithm for such a D is the same as that presented in the previous section, Algorithm 1.

It is easy to check the 1d fused lasso matrix is diagonally dominant, as both the left- and right-hand sides of the inequality in (23) are equal to 2 when $D = D_{1d}$. Unfortunately, neither the 2d fused lasso matrix nor any of the trend filtering matrices satisfy condition (23). In fact, examples show that the boundary lemma does not hold for these cases. However, inspired by the 1d fused lasso, we can develop a similar strategy to compute the full solution path for an arbitrary matrix D. The difference is: in addition to checking when coordinates will hit the boundary, we have to check when coordinates will leave the boundary as well.

6.1 Path algorithm

Recall that we defined, at a particular λ_k, the "hitting time" of an interior coordinate path to the value of $\lambda \leq \lambda_k$ at which this path hits the boundary. Similarly, let us define the "leaving time" of a boundary coordinate path to be the value of $\lambda \leq \lambda_k$ at which this path leaves the boundary (we will make this idea more precise shortly). We call the coordinate with the largest hitting time the "hitting coordinate", and the one with the largest leaving time the "leaving coordinate". As before, we maintain a list $\mathcal{B}$ of boundary coordinates, and s contains their signs. The algorithm for a general matrix D is:

Algorithm 2 (Dual path algorithm for a general D).

- *Start with $k = 0$, $\lambda_0 = \infty$, $\mathcal{B} = \emptyset$, and $s = \emptyset$.*
- *While $\lambda_k > 0$:*
    1. *Compute a solution at λ_k by least squares, as in* (26).

2. *Compute the next hitting time h_{k+1}, as in* (27) *and* (28).
3. *Compute the next leaving time l_{k+1}, as in* (29), (30), *and* (31).
4. *Set $\lambda_{k+1} = \max\{h_{k+1}, l_{k+1}\}$. If $h_{k+1} > l_{k+1}$ then add the hitting coordinate to $\mathcal{B}$ and its sign to s, otherwise remove the leaving coordinate from $\mathcal{B}$ and its sign from s. Set $k = k+1$.*

Although the intuition for this algorithm comes from the 1d fused lasso problem, its details are derived from a more technical point of view, via the Karush-Kuhn-Tucker (KKT) optimality conditions. For our problem (13), the KKT conditions are

$$DD^T\hat{u}_\lambda - Dy + \alpha\gamma = 0, \tag{24}$$

where $\hat{u}_\lambda, \alpha, \gamma$ are subject to the constraints

$$\|\hat{u}_\lambda\|_\infty \le \lambda \tag{25a}$$
$$\alpha \ge 0 \tag{25b}$$
$$\alpha \cdot (\|\hat{u}_\lambda\|_\infty - \lambda) = 0 \tag{25c}$$
$$\|\gamma\|_1 \le 1 \tag{25d}$$
$$\gamma^T\hat{u}_\lambda = \|\hat{u}_\lambda\|_\infty. \tag{25e}$$

Constraints (25d), (25e) say that γ is a subgradient of the function $x \mapsto \|x\|_\infty$ evaluated at $x = \hat{u}_\lambda$. Subgradients are a generalization of gradients to the case of nondifferentiable functions—for an overview, see [2].

A necessary and sufficient condition for $\hat{u}_\lambda$ to be a solution to (13) is that $\hat{u}_\lambda, \alpha, \gamma$ satisfy (24) and (25a)–(25e) for some α and γ. The basic idea is that hitting times are events in which (25a) is violated, and leaving times are events in which (25b)–(25e) are violated. We now describe what happens at the kth iteration. At $\lambda = \lambda_k$, the path's solution is $\hat{u}_{\lambda_k,\mathcal{B}} = \lambda_k s$ for the boundary coordinates, and the least squares estimate

$$\hat{u}_{\lambda_k,-\mathcal{B}} = \big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^+ D_{-\mathcal{B}}\big(y - \lambda_k (D_{\mathcal{B}})^T s\big) \tag{26}$$

for the interior coordinates. Here A^+ denotes the (Moore-Penrose) pseudoinverse of a matrix A, which is needed as D may not have full row rank. Write $\hat{u}_{\lambda_k,-\mathcal{B}} = a - \lambda_k b$. Like the 1d fused lasso case, we decrease λ and continue in a linear direction from the interior solution at λ_k, proposing $\hat{u}_{\lambda,-\mathcal{B}} = a - \lambda b$. We first determine when a coordinate of $a - \lambda b$ will hit the boundary. Setting $a_i - \lambda b_i = \pm\lambda$ and solving for λ gives the hitting time of the ith interior coordinate,

$$t_i^{(\text{hit})} = \frac{a_i}{b_i \pm 1} = \frac{\Big[\big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^+ D_{-\mathcal{B}}y\Big]_i}{\Big[\big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^+ D_{-\mathcal{B}}(D_{\mathcal{B}})^T s\Big]_i \pm 1}. \tag{27}$$

(Only one of $+1$ or -1 will yield a value in $[0, \lambda_k]$.[1]) Hence the next hitting time is

$$h_{k+1} = \max_i t_i^{(\text{hit})}. \tag{28}$$

The new step is to determine when a boundary coordinate will next leave the boundary. After examining the constraints (25b)–(25e), we first define

$$c_i = s_i \cdot \bigg[D_{\mathcal{B}}\Big[I - (D_{-\mathcal{B}})^T\big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^+ D_{-\mathcal{B}}\Big]y\bigg]_i \tag{29}$$
$$d_i = s_i \cdot \bigg[D_{\mathcal{B}}\Big[I - (D_{-\mathcal{B}})^T\big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^+ D_{-\mathcal{B}}\Big](D_{\mathcal{B}})^T s\bigg]_i,$$

[1]If i is the coordinate that left the boundary in the last iteration, then the choice of ±1 here is actually determined by the sign of the boundary opposite to the one it left.

and then we can express the leaving time of the ith boundary coordinate as

$$t_i^{(\text{leave})} = \begin{cases} c_i/d_i & \text{if } c_i < 0 \text{ and } d_i < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

Therefore the next leaving time is

$$l_{k+1} = \max_i t_i^{(\text{leave})}. \tag{31}$$

The last step of the iteration moves until the next critical event—hitting time or leaving time, whichever happens first. We can verify that the path visited by the algorithm satisfies the KKT conditions (24) and (25a)–(25e) at each λ, and hence is indeed a solution path of the dual problem (13). This argument, as well a derivation of the leaving times given in (29) and (30), can be found in the online supplement.

6.2 Properties of the solution path

Suppose that the algorithm terminates after T iterations. By construction, the returned solution path $\hat{u}_\lambda$ is piecewise linear with respect to λ, with kinks at $\lambda_1 \geq \ldots \geq \lambda_T$. Continuity, on the other hand, is a little more subtle: because of the specific choice of the pseudoinverse solution in (26), the path $\hat{u}_\lambda$ is continuous over λ. (When A does not have full column rank, there are many minimizers of $\|z - Ax\|_2$, and $x = (A^TA)^+A^Tz$ is only one of them.) The proof of continuity appears in the online supplement.

Since the primal solution path $\hat{\beta}_\lambda$ can be recovered from $\hat{u}_\lambda$ by the linear transformation (14), the path $\hat{\beta}_\lambda$ is also continuous and piecewise linear in λ. The kinks in this path are necessarily a subset of $\{\lambda_1, \ldots \lambda_T\}$. However, this could be a strict inclusion as $\mathrm{rank}(D)$ could be $< m$, that is, D^T could have a nontrivial null space. So when does the primal solution path change slope? To answer this question, it helps to write the solutions in a more explicit form.

For any given λ, let $\mathcal{B} = \mathcal{B}(\lambda)$ and $s = s(\lambda)$ be the current boundary coordinates and their signs. Then we know that the dual solution can be written as

$$\begin{aligned} \hat{u}_{\lambda,\mathcal{B}} &= \lambda s \\ \hat{u}_{\lambda,-\mathcal{B}} &= \big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^+ D_{-\mathcal{B}}\big(y - \lambda(D_\mathcal{B})^T s\big). \end{aligned}$$

This means that the dual fit $D^T\hat{u}_\lambda$ is just

$$D^T\hat{u}_\lambda \;=\; (D_\mathcal{B})^T\hat{u}_{\lambda,\mathcal{B}} + (D_{-\mathcal{B}})^T\hat{u}_{\lambda,-\mathcal{B}} \;=\; \lambda(D_\mathcal{B})^T s + P_{\mathrm{row}(D_{-\mathcal{B}})}\big(y - \lambda(D_\mathcal{B})^T s\big), \tag{32}$$

where P_M denotes the projection operator onto a linear subspace M (here the row space of $D_{-\mathcal{B}}$). Therefore, applying (14), the primal solution is given by

$$\hat{\beta}_\lambda \;=\; \big(I - P_{\mathrm{row}(D_{-\mathcal{B}})}\big)\big(y - \lambda(D_\mathcal{B})^T s\big) \;=\; P_{\mathrm{null}(D_{-\mathcal{B}})}\big(y - \lambda(D_\mathcal{B})^T s\big). \tag{33}$$

Equation (33) is useful for several reasons. Later, in Section 10, we use it along with a geometric argument to prove a result on the degrees of freedom of $\hat{\beta}_\lambda$. But first, equation (33) can be used to answer our immediate question about the primal path's changes in slope: it turns out that $\hat{\beta}_\lambda$ changes slope at λ_{k+1} if $\mathrm{null}(D_{-\mathcal{B}(\lambda_k)}) \neq \mathrm{null}(D_{-\mathcal{B}(\lambda_{k+1})})$, that is, the null space of $D_{-\mathcal{B}}$ changes from iterations k to $k+1$. (The proof is left to the online supplement.) Thus we have achieved a generalization of the primal-dual correspondence of Section 5.3:

- *Primal-dual correspondence for a general D.* The values of λ at which at which the primal coordinates changes slope are the values of λ at which the null space of $D_{-\mathcal{B}(\lambda)}$ changes.

For various applications, the null space of $D_{-\mathcal{B}}$ can have a nice interpretation. We present the case for the fused lasso on an arbitrary graph $\mathcal{G}$, with m edges and n nodes. We assume without a loss of generality that $\mathcal{G}$ is connected (otherwise the problem decouples into smaller fused lasso problems). Recall that in this setting each row of D gives the difference between two nodes connected by an edge. Hence the null space of D is spanned by the vector of all ones

$$\mathbb{1} = (1, 1, \ldots 1)^T \in \mathbb{R}^n.$$

Furthermore, removing a subset of the rows, as in $D_{-\mathcal{B}}$, is like removing the corresponding subset of edges, yielding a subgraph $\mathcal{G}_{-\mathcal{B}}$. It is not hard to see that the dimension of the null space of $D_{-\mathcal{B}}$ is equal to the number of connected components in $\mathcal{G}_{-\mathcal{B}}$. In fact, if $\mathcal{G}_{-\mathcal{B}}$ has connected components $A_1, \ldots A_k$, then the null space of $D_{-\mathcal{B}}$ is spanned by $\mathbb{1}_{A_1}, \ldots \mathbb{1}_{A_k} \in \mathbb{R}^m$, the indicator vectors on these components,

$$(\mathbb{1}_{A_i})_j = 1(\text{node } j \in A_i) \quad \text{for} \quad j = 1, \ldots n.$$

When $\mathcal{G}_{-\mathcal{B}}$ has connected components $A_1, \ldots A_k$, the projection $P_{\text{null}(D_{-\mathcal{B}})}$ performs a coordinate-wise average within each group A_i:

$$P_{\text{null}(D_{-\mathcal{B}})}(x) = \sum_{i=1}^{k} \left(\frac{(\mathbb{1}_{A_i})^T x}{|A_i|} \right) \cdot \mathbb{1}_{A_i}.$$

Therefore, recalling (33), we see that coordinates of the primal solution $\hat{\beta}_\lambda$ are constant (or in other words, fused) on each group A_i.

As λ decreases, the boundary set $\mathcal{B}$ can both grow and shrink in size; this corresponds to adding an edge to and removing an edge from the graph $\mathcal{G}_{-\mathcal{B}}$, respectively. Since the null space of $D_{-\mathcal{B}}$ can only change when $\mathcal{G}_{-\mathcal{B}}$ undergoes a change in connectivity, the general primal-dual correspondence stated above becomes:

- *Primal-dual correspondence for the fused lasso on a graph.* In two parts:
  - (i) the values of λ at which two primal coordinate groups fuse are the values of λ at which a dual coordinate hits the boundary and disconnects the graph $\mathcal{G}_{-\mathcal{B}(\lambda)}$;
  - (ii) the values of λ at which two primal coordinate groups un-fuse are the values of λ at which a dual coordinate leaves the boundary and reconnects the graph $\mathcal{G}_{-\mathcal{B}(\lambda)}$.

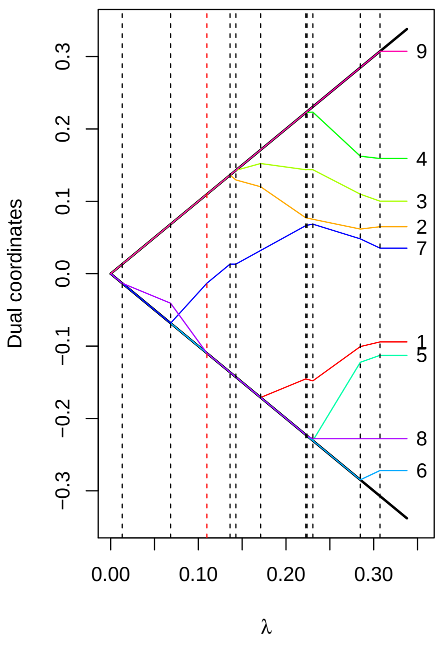
Figure 8 illustrates this correspondence for a graph with $n = 6$ nodes and $m = 9$ edges. Note that the primal-dual correspondence for the fused lasso on a graph, as stated above, is consistent with that given in Section 5.3. This is because the 1d fused lasso corresponds to a chain graph, so removing an edge always disconnects the graph, and furthermore, no dual coordinates ever leave the boundary by the boundary lemma.

7 A general design matrix X

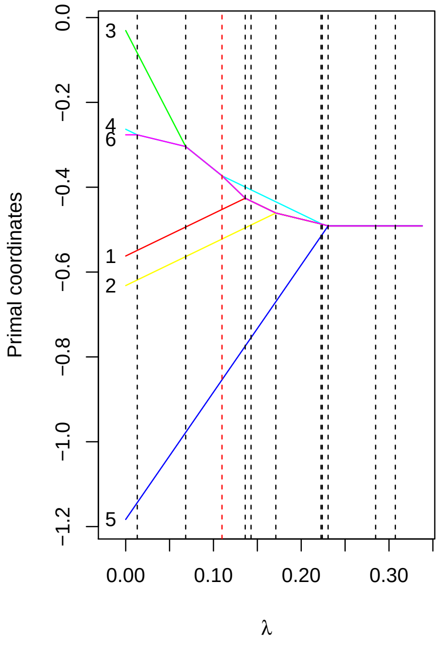
In the last two sections, we focused on the signal approximation case $X = I$. In this section we consider the problem (2) when X is a general $n \times p$ matrix of predictors (and D is a general $m \times p$ penalty matrix). Our strategy is to again solve the equivalent dual problem (17). At first glance this problem looks much more difficult than the dual (13) when $X = I$. Moreover, the relationship between the primal and dual solutions is now

$$\hat{\beta}_\lambda = (X^T X)^+ (X^T y - D^T \hat{u}_\lambda) + b, \tag{34}$$

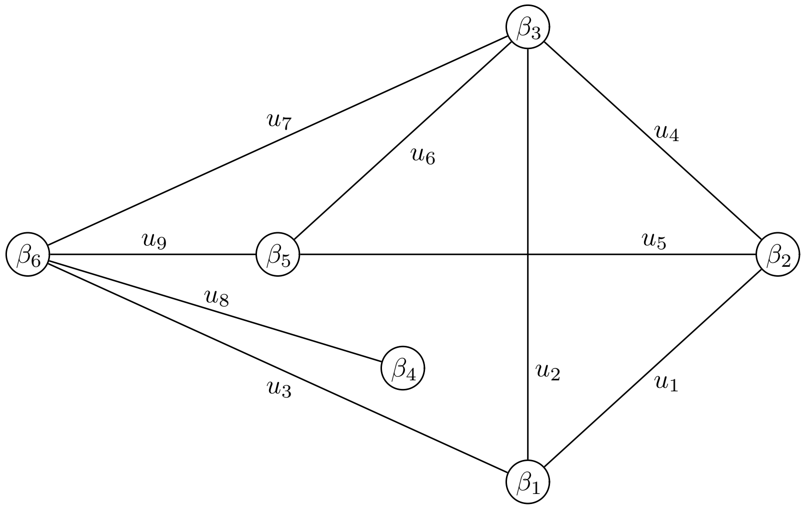
where $b \in \text{null}(X)$, which is also more complicated.



(a) Dual



(b) Primal



(c) Underlying graph

Figure 8: *Dual and primal coordinate paths for the fused lasso applied to the shown graph structure. As λ decreases, the first dual coordinate to hit the boundary is u_9, but removing the corresponding edge doesn't disconnect the graph, so nothing happens in the primal setting. Then u_6 hits the boundary, and again, removing its edge doesn't affect the graph's connectivity, so nothing happens. But when u_5 hits the boundary next, removing its edge disconnects the graph (the node marked β_5 becomes its own connected component), and hence two primal coordinate paths fuse. Note that u_8 leaves the boundary at some point (the red dashed vertical line). Adding its edge reconnects the graph, and therefore two primal coordinates un-fuse.*

It turns out that we can make (17) look more familiar—that is, more like (13)—by defining

$$\tilde{y} = XX^+ y \;\text{ and }\; \widetilde{D} = DX^+. \tag{35}$$

(Here the pseudoinverse of a rectangular matrix A is $A^+ = (A^T A)^+ A^T$.) Abbreviating $P = P_{\mathrm{col}(X)} = XX^+$, the objective function in (17) becomes

$$\begin{aligned}(X^T y - D^T u)^T (X^T X)^+ (X^T y - D^T u) &= y^T P y - 2 y^T \widetilde{D}^T u + u^T \widetilde{D}\widetilde{D}^T u \\ &= (y - \widetilde{D}^T u)^T P (y - \widetilde{D}^T u) \\ &= (y - \widetilde{D}^T u)^T P^2 (y - \widetilde{D}^T u) \\ &= (\tilde{y} - \widetilde{D}^T u)^T (\tilde{y} - \widetilde{D}^T u).\end{aligned}$$

The first equality above is by the definition of $\widetilde{D}$; the second holds because $P\widetilde{D}^T = \widetilde{D}^T$; the third is because P is idempotent; and the fourth is again due to the identity $P\widetilde{D}^T = \widetilde{D}^T$. Therefore we can rewrite the dual problem (17) in terms of our transformed data and penalty matrix:

$$\begin{aligned}\underset{u\in\mathbb{R}^m}{\text{minimize}} \;\; & \frac{1}{2}\|\tilde{y} - \widetilde{D}^T u\|_2^2 \\ \text{subject to} \;\; & \|u\|_\infty \le \lambda, \; D^T u \in \mathrm{row}(X),\end{aligned} \tag{36}$$

It is also helpful to rewrite the relationship (34) in terms of our new data and penalty matrix:

$$\hat{\beta}_\lambda = X^+ (\tilde{y} - \widetilde{D}^T \hat{u}_\lambda) + b, \tag{37}$$

where $b \in \mathrm{null}(X)$, which implies that the fit is simply

$$X\hat{\beta}_\lambda = \tilde{y} - \widetilde{D}^T \hat{u}_\lambda. \tag{38}$$

Modulo the row space constraint, $D^T u \in \mathrm{row}(X)$, problem (36) has exactly the same form as the dual (13) studied in Section 6. In the case that X has full column rank, this extra constraint has no effect, so we can treat the problem just as before. We discuss this next.

7.1 The case $\mathrm{rank}(X) = p$

Suppose that $\mathrm{rank}(X) = p$, so $\mathrm{row}(X) = \mathbb{R}^p$ (note that this necessarily means $p \le n$). Hence we can remove the constraint $D^T u \in \mathrm{row}(X)$ from problem (36), so that it is really just the same as problem (13) that we solved in Section 6, except with y, D replaced by $\tilde{y}, \widetilde{D}$, respectively. Therefore we can apply Algorithm 2 to find a dual solution path $\hat{u}_\lambda$. This gives the (unique) primal solution path using (37) with $b = 0$ (since $\mathrm{null}(X) = \{0\}$), or the fit using (38).

Fortunately, all of the properties in Section 6.2 apply to the current setting as well. First, we know that the constructed dual path $\hat{u}_\lambda$ is continuous and piecewise linear, because we are using the same algorithm as before, just with a different data vector and penalty matrix. This means that $\hat{\beta}_\lambda$ is also continuous and piecewise linear, since it is given by the linear transformation (37). Next, we can follow the same logic in writing out the dual fit $\widetilde{D}^T \hat{u}_\lambda$ to conclude that

$$\hat{\beta}_\lambda = X^+ P_{\mathrm{null}(\widetilde{D}_{-\mathcal{B}})} \big(\tilde{y} - \lambda (\widetilde{D}_{-\mathcal{B}})^T s\big), \tag{39}$$

or

$$X\hat{\beta}_\lambda = P_{\mathrm{null}(\widetilde{D}_{-\mathcal{B}})} \big(\tilde{y} - \lambda (\widetilde{D}_{-\mathcal{B}})^T s\big). \tag{40}$$

Hence $0 = \widetilde{D}_{-\mathcal{B}} X \hat{\beta}_\lambda = D_{-\mathcal{B}} \hat{\beta}_\lambda$, which means that $\hat{\beta}_\lambda \in \mathrm{null}(D_{-\mathcal{B}})$, as before.

Though working with equations (39) and (40) may seem complicated (as one would need to expand the newly defined variables $\tilde{y}, \widetilde{D}$ in terms of y, D), it is straightforward to show that the general primal-dual correspondence still holds here. This is given in the online supplement. That is: the primal path $\hat{\beta}_\lambda$ changes slope at the values of λ at which the null space of $D_{-\mathcal{B}(\lambda)}$ changes. For the fused lasso on a graph $\mathcal{G}$, we indeed still get fused groups of coordinates in the primal solution, since $\hat{\beta}_\lambda \in \mathrm{null}(D_{-\mathcal{B}})$ implies that $\hat{\beta}_\lambda$ is fused on the connected components of $\mathcal{G}_{-\mathcal{B}}$. Therefore, fusions still correspond to dual coordinates hitting the boundary and disconnecting the graph, and un-fusions still correspond to dual coordinates leaving the boundary and reconnecting the graph.

7.2 The case $\mathrm{rank}(X) < p$

If $\mathrm{rank}(X) < p$, then $\mathrm{row}(X)$ is a strict subspace of $\mathbb{R}^p$. One easy way to avoid dealing with the constraint $D^T u \in \mathrm{row}(X)$ of (36) is to add an ℓ_2 penalty to our original problem, giving

$$\underset{\beta \in \mathbb{R}^p}{\mathrm{minimize}} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \|D\beta\|_1 + \frac{\epsilon}{2}\|\beta\|_2^2, \tag{41}$$

where $\epsilon > 0$ is fixed. This is analogous to the *elastic net*, which adds an ℓ_2 penalty to the lasso criterion [34]. Problem (41) can be rewritten as

$$\underset{\beta \in \mathbb{R}^p}{\mathrm{minimize}} \ \frac{1}{2}\|y^* - (X^*)\beta\|_2^2 + \lambda \|D\beta\|_1,$$

where $y^* = (y, 0)^T$ and $X^* = \begin{bmatrix} X \\ \sqrt{\epsilon} I \end{bmatrix}$. Since $\mathrm{rank}(X^*) = p$, we can now use the strategy discussed in the last section, which is just applying Algorithm 2 to a transformed problem, to find the solution path of (41). Putting aside computational concerns, it may still be preferable to study problem (41) instead of the original generalized lasso problem (2). Some reasons are:

- the (unique) solution path of (41) may be more stable than the (non-unique) generalized lasso path, analogous to the behavior of the elastic net and lasso paths;
- the fit of (41) may actually outperform the generalized lasso fit in terms prediction error, again like the comparison between the elastic net and lasso prediction errors.

Though adding an ℓ_2 penalty is easier and, as we suggested, perhaps even desirable, we can still solve the unmodified problem (2) in the $\mathrm{rank}(X) < p$ case, by looking at its dual (36). We only give a rough sketch of the path algorithm because in the present setting the solution and its computation are more complicated.

We can rewrite the row space constraint in (36) as $D^T u \perp \mathrm{null}(X)$. Write $P = P_{\mathrm{null}(X)}$, and then problem (36) is

$$\begin{aligned} &\underset{u \in \mathbb{R}^m}{\mathrm{minimize}} \ \frac{1}{2}\|\tilde{y} - \widetilde{D}^T u\|_2^2 \\ &\text{subject to } \ \|u\|_\infty \le \lambda, \ (DP)^T u = 0. \end{aligned} \tag{42}$$

To find a solution path of (42), the KKT conditions (24) and (25a)-(25e) need to be modified to incorporate the new equality constraint. These are now

$$\widetilde{D}\widetilde{D}^T \hat{u}_\lambda - \widetilde{D}\tilde{y} + \alpha\gamma + DP\delta = 0,$$

where $\hat{u}_\lambda, \alpha, \gamma, \delta$ are subject to the same constraints as before, (25a)–(25e), and additionally $(DP)^T \hat{u}_\lambda = 0$. Instead of simply using the appropriate least squares estimate at each iteration, we now need to solve for $\hat{u}_\lambda$ and δ jointly. When $\lambda = \infty$, this can be done by solving the block system

$$\begin{bmatrix} \widetilde{D}\widetilde{D}^T & DP \\ (DP)^T & 0 \end{bmatrix} \begin{bmatrix} \hat{u}_\lambda \\ \delta \end{bmatrix} = \begin{bmatrix} \widetilde{D}\tilde{y} \\ 0 \end{bmatrix}, \tag{43}$$

and in future iterations the expressions are similar. Having done this, satisfying the rest of the constraints (25a)–(25e) can be done by finding the hitting and leaving times just as we did previously.

8 Computational considerations

Here we discuss an efficient implementation of Algorithm 2, which gives the solution path of the signal approximation problem (12), after applying the transformation (14) from dual to primal variables. For a design with $\mathrm{rank}(X) = p$, we can modify y and X as in (35), and then the same algorithm gives the solution path of (2), this time relying on the transformation (37) (with $b = 0$) for the primal path.

At each iteration of the algorithm, the dominant work is in computing expressions of the form

$$\big(D_{-\mathcal{B}}(D_{-\mathcal{B}})^T\big)^+ D_{-\mathcal{B}}x$$

for a vector $x \in \mathbb{R}^n$, where $\mathcal{B}$ is the current boundary set (see equations (27) and (29)). Equivalently, the complexity of each iteration is based on finding

$$\underset{v}{\mathrm{argmin}} \left\{ \|v\|_2 \ : \ v \in \underset{w}{\mathrm{argmin}} \, \|x - (D_{-\mathcal{B}})^T w\|_2 \right\}, \tag{44}$$

the least squares solution with the smallest ℓ_2 norm. In the next iteration, $D_{-\mathcal{B}}$ has either one less or one more row (depending on whether a coordinate hit or left the boundary).

We can exploit the fact that the problems (44) are highly related from one iteration to the next (our strategy that is similar to that in the LARS implementation). Suppose that when $\mathcal{B} = \emptyset$, we solve the problem (44) by using a matrix factorization (for example, a QR decomposition). In future iterations, this factorization can be efficiently updated after a row has been deleted from or added to $D_{-\mathcal{B}}$. This allows us to compute the new solution of (44) with much less work than it would take to solve the problem from "scratch".

Recall that D is $m \times n$, and the dual variable u is m-dimensional. Let T denote the number of iterations taken by the algorithm (note that $T \geq m$, and can be strictly greater if dual coordinates leave the boundary). When $m \leq n$, we can use a QR decomposition of D^T to compute the full dual solution path in

$$O(mn^2 + Tm^2)$$

operations. When $m > n$, using a QR decomposition of D allows us to compute the full dual solution path in

$$O(m^2 n + Tn^2)$$

operations. The main idea behind this implementation is fairly straightforward. However, the details are somewhat complicated because we require the minimum ℓ_2 norm solution (44), instead of a generic solution, to the appropriate least squares problem at each iteration. See Chapters 5 and 12 of [15] for an extensive coverage of the QR decomposition.

We mention two simple points to improve practical efficiency:

- The algorithm starts at the fully regularized end of the path ($\lambda = \infty$) and works towards the un-regularized solution ($\lambda = 0$). Therefore, for problems in which the highly or moderately regularized solutions are the only ones of interest, the algorithm can compute only part of the path and terminate early. This could end up being a large savings in practice.

- One can obtain an approximate solution path by not permitting dual coordinates to leave the boundary (achieved by setting $l_{k+1} = 0$ in Step 3 of Algorithm 2). This makes $T = m$, and so computing this approximate path only requires $O(mn^2)$ or $O(m^2 n)$ operations when $m \leq n$ or $m > n$, respectively. This approximation can be quite accurate if the number times a dual coordinate leaves the boundary is (relatively) small. Furthermore, its legitimacy is supported by the following fact: for $D = I$ (the lasso problem) and $\mathrm{rank}(X) = p$, this approximate path is exactly the LARS path when LARS is run in its original (unmodified) state. We discuss this in the next section.

Finally, it is important to note that if one's goal is to find the solution of (12) or (2) over a discrete set of λ values, and the problem size is very large, then it is unlikely that our path algorithm is the most efficient approach. A reason here is the same reason that LARS is not generally used to solve large-scale lasso problems: the set of critical points (changes in slope) in the piecewise linear solution path $\hat{\beta}_\lambda$ becomes very dense as the problem size increases. But there is a further disadvantage is that is specific to the generalized lasso (and not the lasso). At the regularized end of the path (which is typically the region of interest), the generalized lasso solution is orthogonal to many of the rows of D; therefore we must solve large least squares problems, as in (26), in order to compute the path. On the other hand, at the regularized end of the lasso path, the solution lies in the span of a small number of columns of X; therefore we only need to solve small least squares problems to compute the path.

For solving a large generalized lasso (or lasso) problem at a fixed λ, it is preferable to use a convex optimization technique that was specifically developed for the purposes of computational efficiency. First-order methods, for example, can efficiently solve large-scale instances of (12) or (2) for λ in a discrete set (see [1] as an example). Another optimization method of recent interest is coordinate descent [33], which is quite efficient in solving the lasso at discrete values of λ [14], and is favored for its simplicity. But coordinate descent cannot be used for the minimizations (12) and (2), because the penalty term $\|D\beta\|_1$ is not separable in β, and therefore coordinate descent does not necessarily converge. In the important signal approximation case (12), however, the dual problem (13) is separable, so coordinate descent will indeed converge if applied to the dual. Moreover, for various applications, the matrix D is sparse and structured, which means that the coordinate-wise updates for (13) are very fast. This makes coordinate descent on the dual a promising method for solving many of the signal approximation problems from Section 2.1.

9 Connection to LARS

In this section, we return to the LARS algorithm, described in the introduction as a point of motivation for our work. Here we assume that $D = I$, so that (2) is the same as the standard lasso problem (1); furthermore we assume that $\mathrm{rank}(X) = p$. Our algorithm computes the lasso path $\hat{\beta}_\lambda$, via the dual path $\hat{u}_\lambda$ (using Algorithm 2 applied to $\tilde{y}, \widetilde{D}$ as defined in (35)). Another way of finding the lasso path is to use the LARS algorithm in its "lasso" mode. Since the problem is strictly convex (X has full column rank), there is only one solution at each λ, so of course these two algorithms must give the same result.

In its original or unmodified state, LARS returns a different path, obtained by selecting variables in order to continuously decrease the maximal absolute correlation with the residual (technically these are inner products with the residual, but they become correlations if we think of standardized data). We refer to this as the "LARS path". Interestingly, the LARS path can be viewed as an approximation to the lasso path (see [11] for an elegant interpretation and discussion of this). In our framework, we can obtain an approximate dual solution path if we never check for dual coordinates leaving the boundary, which can be achieved by dropping Step 3 from Algorithm 2 (or more precisely, by setting $l_{k+1} = 0$ for each k). If we denote the resulting dual path by $\tilde{u}_\lambda$, then this suggests a primal path

$$\tilde{\beta}_\lambda = (X^T X)^{-1}(X^T y - \tilde{u}_\lambda), \tag{45}$$

based on the transformation (34) (noting that $b = 0$ as $\mathrm{null}(X) = \{0\}$). The question is: how does this approximate solution path $\tilde{\beta}_\lambda$ compare to the LARS path?

Figure 9 shows the two paths in question. On the left is the familiar plot of [11], showing the LARS path for the "diabetes data". The colored dots on the x-axis mark when variables enter the model. The right plot shows our approximate solution path on this same data set, with vertical dashed lines marking when variables (coordinates) hit the boundary. The paths look identical, and this is not a coincidence: we show that our approximate path, which is given by ignoring dual coordinates leaving the boundary, is equal to the LARS path in general.

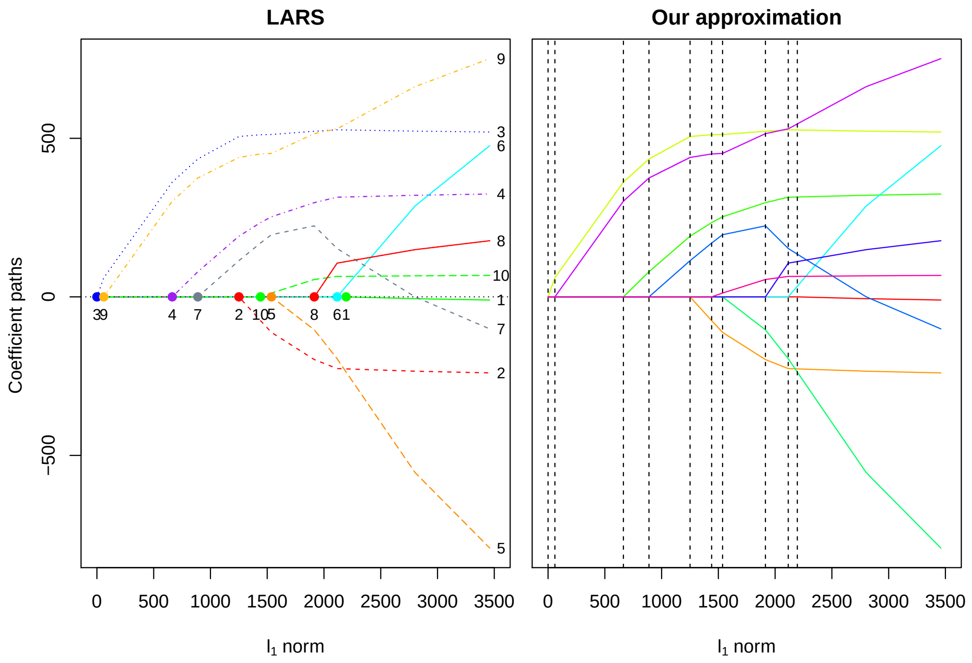


Figure 9: *Comparing the LARS path and our approximate lasso path, on the diabetes data. For this data set* $n = 442$ *and* $p = 10$. *The paths by parametrized by the* ℓ_1 *norm of their (respective) coefficient vectors.*

Lemma 2 (Equivalence to LARS). *Suppose that* $\operatorname{rank}(X) = p$ *and consider using Algorithm 2 to compute an approximate lasso path in the following way: we use* $\tilde{y} = XX^+y$, $\widetilde{D} = X^+$ *in place of* y, D, *and we ignore Step 3 (that is, set* $l_{k+1} = 0$*). Let* $\tilde{u}_\lambda$ *denote the corresponding dual path, and define a primal path* $\tilde{\beta}_\lambda$ *according to (45). Then* $\tilde{\beta}_\lambda$ *is exactly the LARS path.*

Proof. First define the residual $r_\lambda = y - X\tilde{\beta}_\lambda$. Notice that by rearranging (45), we get $\tilde{u}_\lambda = X^T r_\lambda$. Therefore, the coordinates of the dual path are equal to the correlations (inner products) of the columns of X with the current residual. Hence we have a procedure that:

- moves in a direction so that the absolute correlation with the current residual is constant within $\mathcal{B}$ (and maximal among all variables) for all λ;
- adds variables to $\mathcal{B}$ once their absolute correlation with the residual matches that realized in $\mathcal{B}$.

This almost proves that $\tilde{\beta}_\lambda$ is the LARS path, with $\mathcal{B}$ being the "active set" in LARS terminology. What remains to be shown is that the variables not in $\mathcal{B}$ are all assigned zero coefficients. But, recalling that $D = I$, the same arguments given in Section 6.2 and Section 7.1 apply here to give that $\tilde{\beta}_\lambda \in \operatorname{null}(I_{-\mathcal{B}})$ (really, $\tilde{u}_\lambda$ still solves a sequence of least squares problems, and the only difference between $\tilde{u}_\lambda$ and $\hat{u}_\lambda$ is in how we construct $\mathcal{B}$). This means that $\tilde{\beta}_{\lambda,-\mathcal{B}} = 0$, as desired. $\square$

10 Degrees of freedom

In applied statistics, degrees of freedom describes the effective number of parameters used by a fitting procedure. This is typically easy to compute for linear procedures (linear in the data y) but difficult for nonlinear, adaptive procedures. In this section, we derive the degrees of freedom of the fit of the generalized lasso problem, when $\mathrm{rank}(X) = p$ and D is an arbitrary penalty matrix. This produces corollaries on degrees of freedom for various problems presented in Section 2. We then briefly discuss model selection using these degrees of freedom results, and lastly we discuss the role of shrinkage, a fundamental property of ℓ_1 regularization.

10.1 Degrees of freedom results

We assume that the data vector $y \in \mathbb{R}^n$ is drawn from the normal model

$$y \sim N(\mu, \sigma^2 I),$$

and the design matrix X is fixed (nonrandom). For a function $g : \mathbb{R}^n \to \mathbb{R}^n$, with ith coordinate function $g_i : \mathbb{R}^n \to \mathbb{R}$, the degrees of freedom of g is defined as

$$\mathrm{df}(g) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}\big(g_i(y), y_i\big).$$

For our problem, the function of interest is $g(y) = X\hat{\beta}_\lambda(y)$, for fixed λ.

An alternative and convenient formula for degrees of freedom comes from Stein's unbiased risk estimate [30]. If g is continuous and almost differentiable, then Stein's formula states that

$$\frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}\big(g_i(y), y_i\big) = \mathrm{E}[(\nabla \cdot g)(y)]. \tag{46}$$

Here $\nabla \cdot g = \sum_{i=1}^n \partial g_i / \partial y_i$ is called the divergence of θ. This is useful because typically the right-hand side of (46) is easier to calculate; for our problem this is the case. But using Stein's formula requires checking that the function is continuous and almost differentiable. In addition to checking these regularity conditions for $g(y) = X\hat{\beta}_\lambda(y)$, we establish below that for almost every y the fit $X\hat{\beta}_\lambda(y)$ is a locally affine projection. Essentially, this allows us to take the divergence in (33) when $X = I$, or (40) for the general X case, and treat $\mathcal{B}$ and s as constants.

As in our development of the path algorithm in Sections 5, 6, and 7, we first consider the case $X = I$, because it is easier to understand. Notice that we can express the dual fit as $D^T\hat{u}_\lambda(y) = P_{C_\lambda}(y)$, the projection of y onto the convex polytope

$$C_\lambda = \{D^T u : \|u\|_\infty \le \lambda\} \subseteq \mathbb{R}^n.$$

From (14), the primal solution is just the residual from this projection, $\hat{\beta}_\lambda(y) = (I - P_{C_\lambda})(y)$. The projection map onto a convex set is always nonexpansive, that is, Lipschitz continuous with constant 1. In fact, so is the residual from projecting onto a convex set (for example, see the proof of Theorem 1.2.2 in [27]). Therefore $\hat{\beta}_\lambda(y)$ is nonexpansive in y, and hence continuous and almost differentiable (this follows from the standard proof a result called "Rademacher's theorem"; for example, see Theorem 2 in Section 3.2 of [13]).

Furthermore, thinking geometrically about the projection map onto C_λ yields a crucial insight. Examine Figure 10—as drawn, it is clear that we can move the point y slightly and it still projects to the same face of C_λ. In fact, it seems that the only points y for which this property does not hold necessarily lie on rays that emanate orthogonally from the corners of C_λ (two such rays are drawn leaving the bottom right corner). In other words, we are lead to believe that for almost every y, the projection map onto C_λ is a locally constant affine projection. This is indeed true:

Lemma 3. *For fixed λ, there exists a set $\mathcal{N}_\lambda$ such that:*

(a) $\mathcal{N}_\lambda$ has Hausdorff dimension $n-1$, hence Lebesgue measure zero;

(b) for any $y \notin \mathcal{N}_\lambda$, there exists a neighborhood U of y such that $P_{C_\lambda} : U \to \mathbb{R}^n$ is simply the projection onto an affine subspace. In particular, the affine subspace is

$$\lambda (D_\mathcal{B})^T s + \mathrm{row}(D_{-\mathcal{B}}), \tag{47}$$

where $\mathcal{B}$ and s are the boundary set and signs for a solution $\hat{u}_\lambda(y)$ of the dual problem (13),

$$\mathcal{B} = \{i : |\hat{u}_{\lambda,i}(y)| = \lambda\} \ \ and \ \ s = \mathrm{sign}\big(\hat{u}_{\lambda,\mathcal{B}}(y)\big).$$

The quantity (47) is well-defined in the sense that it is invariant under different choices of $\mathcal{B}$ and s (as the dual solution may not be unique).

The proof, which follows the intuition described above, is given in the online supplement.

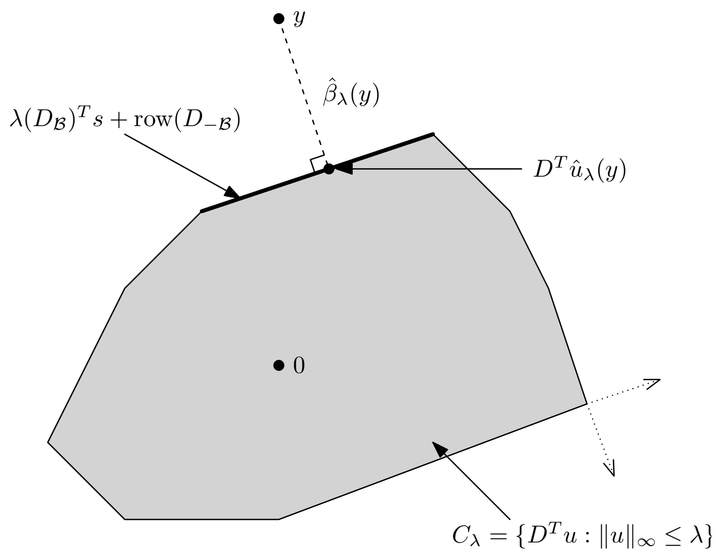

y
$\hat{\beta}_\lambda(y)$
$\lambda(D_\mathcal{B})^T s + \mathrm{row}(D_{-\mathcal{B}})$
$D^T \hat{u}_\lambda(y)$
0
$C_\lambda = \{D^T u : \|u\|_\infty \le \lambda\}$

Figure 10: *An illustration of the geometry surrounding $\hat{u}_\lambda$ and $\hat{\beta}_\lambda$, for the case $X = I$. Recall that $\hat{\beta}_\lambda(y) = y - D^T \hat{u}_\lambda(y)$, where $D^T \hat{u}_\lambda(y)$ is the projection of y onto the convex polytope $C_\lambda = \{D^T u : \|u\|_\infty \le \lambda\}$. Almost everywhere, small perturbations of y don't change the face on which its projection lies. The exceptional set $\mathcal{N}_\lambda$ of points for which this property does not hold has dimension $n-1$, and is a union of rays like the two drawn as dotted lines in the bottom right of the figure.*

Hence we have the following result:

Theorem 1. *For fixed λ, the solution $\hat{\beta}_\lambda$ of the signal approximation problem (12) has degrees of freedom*

$$\mathrm{df}(\hat{\beta}_\lambda) = \mathrm{E}[\mathrm{nullity}(D_{-\mathcal{B}(y)})],$$

where the nullity of a matrix is the dimension of its null space. The expectation here is taken over $\mathcal{B}(y)$, the boundary set of a dual solution $\hat{u}_\lambda(y)$.

Note: above, we can choose any dual solution at y to construct the boundary set $\mathcal{B}(y)$, because by Lemma 3, all dual solutions give rise to the same linear subspace $\mathrm{null}(D_{-\mathcal{B}(y)})$ (almost everywhere in y).

Proof. Consider $y \notin \mathcal{N}_\lambda$, and let $\mathcal{B}$ and s be the boundary set and signs of a dual solution $\hat{u}_\lambda(y)$. By Lemma 3, there is a neighborhood U of y such that

$$\hat{\beta}_\lambda(y') = (I - D^T \hat{u}_\lambda)(y') = P_{\mathrm{null}(D_{-\mathcal{B}})}\big(y' - \lambda (D_{\mathcal{B}})^T s\big)$$

for all $y' \in U$. Taking the divergence at y we get

$$(\nabla \cdot \hat{\beta}_\lambda)(y) = \mathrm{tr}(P_{\mathrm{null}(D_{-\mathcal{B}})}) = \mathrm{nullity}(D_{-\mathcal{B}}),$$

since the trace of a projection matrix is just its rank. This holds for almost every y because $\mathcal{N}_\lambda$ has measure zero, and we can use Stein's formula to conclude that $\mathrm{df}(\hat{\beta}_\lambda) = \mathrm{E}[\mathrm{nullity}(D_{-\mathcal{B}(y)})]$. □

We now consider problem (2), with the predictor matrix satisfying $\mathrm{rank}(X) = p$, and it turns out that the same degrees of freedom formula holds for the fit $X\hat{\beta}_\lambda$. This is relatively straightforward to show, but requires sorting out the details of how to turn statements involving $\tilde{y}, \widetilde{D}$ into those involving y, D. First, by the same arguments as before, we know that $X\hat{\beta}_\lambda(\tilde{y})$ is nonexpansive as a function of $\tilde{y}$. But $\tilde{y} = P_{\mathrm{col}(X)}(y)$ is nonexpansive in y, so $X\hat{\beta}_\lambda(y)$ is indeed nonexpansive, hence continuous and almost differentiable, as a function of y.

Next we must establish that $\widetilde{D}^T \hat{u}_\lambda(y)$ is a locally affine projection for almost every y. Well, by Lemma 3, this is true of $\widetilde{D}^T \hat{u}_\lambda(\tilde{y})$ for $\tilde{y} \notin \mathcal{N}_\lambda$, so we have the desired result except on $\mathcal{M}_\lambda = (P_{\mathrm{col}(X)})^{-1}(\mathcal{N}_\lambda)$. Following the arguments in the proof of Lemma 3, it is not hard to see that $\mathcal{N}_\lambda$ now has dimension $p-1$, so $\mathcal{M}_\lambda$ has measure zero.

With these properties satisfied, we have the general result:

Theorem 2. *Suppose that* $\mathrm{rank}(X) = p$. *For fixed* λ, *the fit* $X\hat{\beta}_\lambda$ *of the generalized lasso* (2) *has degrees of freedom*

$$\mathrm{df}(X\hat{\beta}_\lambda) = \mathrm{E}[\mathrm{nullity}(D_{-\mathcal{B}(y)})],$$

where $\mathcal{B}(y)$ *is the boundary set of a dual solution* $\hat{u}_\lambda(y)$.

Note: as before, we can construct the boundary set $\mathcal{B}(y)$ from any dual solution at y, because by Lemma 3 the quantity $\mathrm{null}(D_{-\mathcal{B}(y)})$ is invariant (almost everywhere in y).

Proof. Let $y \notin \mathcal{M}_\lambda$. We show that $(\nabla \cdot X\hat{\beta}_\lambda)(y) = \mathrm{nullity}(D_{-\mathcal{B}(y)})$, and then applying Stein's formula (along with the fact that $\mathcal{M}_\lambda$ has measure zero) gives the result.

Let $\mathcal{B}$ denote the boundary set of a dual solution $\hat{u}_\lambda(y)$. Then the fit is

$$X\hat{\beta}_\lambda(y) = P_{\mathrm{null}(\widetilde{D}_{-\mathcal{B}})} P_{\mathrm{col}(X)} y + c,$$

where c denotes the terms that have zero derivative with respect to y. Using the fact $\mathrm{null}(X^+) = \mathrm{null}(X^T)$, and that $\mathrm{null}(\widetilde{D}_{-\mathcal{B}}) \supseteq \mathrm{null}(X^+)$,

$$\begin{aligned} P_{\mathrm{null}(\widetilde{D}_{-\mathcal{B}})} P_{\mathrm{col}(X)} &= P_{\mathrm{null}(\widetilde{D}_{-\mathcal{B}})} - P_{\mathrm{null}(\widetilde{D}_{-\mathcal{B}})} P_{\mathrm{null}(X^+)} \\ &= P_{\mathrm{null}(\widetilde{D}_{-\mathcal{B}})} - P_{\mathrm{null}(X^+)}. \end{aligned}$$

Therefore, computing the divergence:

$$\begin{aligned} \big(\nabla \cdot X\hat{\beta}_\lambda\big)(y) &= \mathrm{nullity}(D_{-\mathcal{B}} X^+) - \mathrm{nullity}(X^+) \\ &= \mathrm{nullity}(D_{-\mathcal{B}}), \end{aligned}$$

where the last equality follows because X has full column rank. This completes the proof. □

We saw in Section 6.2 that the null space of D has a nice interpretation for the fused lasso problem. In this case, the theorem also becomes easier to interpret:

Corollary 1 (Degrees of freedom of the fused lasso). *Suppose that* $\mathrm{rank}(X) = p$ *and that* D *corresponds to the fused lasso penalty on an arbitrary graph. Then for fixed* λ*, the fit* $X\hat{\beta}_\lambda$ *of* (2) *has degrees of freedom*

$$\mathrm{df}(X\hat{\beta}_\lambda) = \mathrm{E}[\,\text{number of fused groups in } \hat{\beta}_\lambda(y)\,].$$

Proof. If $\mathcal{G}$ denotes the underlying graph, we showed in Section 6.2 that $\mathrm{nullity}(D_{-\mathcal{B}(y)})$ is the number of connected components in $\mathcal{G}_{-\mathcal{B}(y)}$. We also showed (see Section 7.1 for the extension to a general design X) that the coordinates of $\hat{\beta}_\lambda(y)$ are fused on the connected components of $\mathcal{G}_{-\mathcal{B}(y)}$, giving the result. □

By slightly modifying the penalty matrix, we can also derive the degrees of freedom of the sparse fused lasso:

Corollary 2 (Degrees of freedom of the sparse fused lasso). *Suppose that* $\mathrm{rank}(X) = p$ *and write* $X_i \in \mathbb{R}^p$ *for the ith row of* X*. Consider the sparse fused lasso problem:*

$$\underset{\beta \in \mathbb{R}^p}{\mathrm{minimize}} \ \sum_{i=1}^n (y_i - X_i^T\beta)^2 + \lambda_1 \sum_{i=1}^p |\beta_i| + \lambda_2 \sum_{(i,j)\in E} |\beta_i - \beta_j|, \tag{48}$$

where E *is an arbitrary set of edges between nodes* $\beta_1, \ldots \beta_p$*. Then for fixed* λ_1, λ_2*, the fit* $X\hat{\beta}_{\lambda_1,\lambda_2}$ *of* (48) *has degrees of freedom*

$$\mathrm{df}(X\hat{\beta}_{\lambda_1,\lambda_2}) = \mathrm{E}[\,\text{number of nonzero fused groups in } \hat{\beta}_{\lambda_1,\lambda_2}(y)\,].$$

Proof. We can write (48) in the generalized lasso framework by taking $\lambda = \lambda_2$ and

$$D = \begin{bmatrix} D_{\mathrm{fuse}} \\ \frac{\lambda_1}{\lambda_2} I \end{bmatrix},$$

where D_{fuse} is the fused lasso matrix corresponding to the underlying graph, with each row giving the difference between two nodes connected by an edge.

In Section 6.2, we analyzed the null space of D_{fuse} to interpret the primal-dual correspondence for the fused lasso. A similar interpretation can be achieved with D as defined above. Let $\mathcal{G}$ denote the underlying graph and suppose that it has m edges (and p nodes), so that D_{fuse} is $m \times p$ and D is $(m+p) \times p$. Also, suppose that we decompose the boundary set as $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, where $\mathcal{B}_1$ contains the dual coordinates in $\{1, \ldots m\}$ and $\mathcal{B}_2$ contains those in $\{m+1, \ldots m+p\}$. We can associate the first m coordinates with the m edges, and the last p coordinates with the p nodes. Then the matrix $D_{-\mathcal{B}}$ defines a subgraph $\mathcal{G}_{-\mathcal{B}}$ that can be constructed as follows:

1. delete the edges of $\mathcal{G}$ that correspond to coordinates in $\mathcal{B}_1$, yielding $\mathcal{G}_{-\mathcal{B}_1}$;
2. delete the nodes of $\mathcal{G}_{-\mathcal{B}_1}$ that correspond to coordinates in $-\mathcal{B}_2$, and delete also all connected nodes, yielding $\mathcal{G}_{-\mathcal{B}}$.

It is straightforward to show that the nullity of $D_{-\mathcal{B}}$ is the number of connected components in $\mathcal{G}_{-\mathcal{B}}$. Furthermore, the solution $\hat{\beta}_{\lambda_1,\lambda_2}(y)$ is fused on each connected component of $\mathcal{G}_{-\mathcal{B}}$ and zero in all other coordinates. Applying Theorem 2 gives the result. □

The above corollary proves a conjecture of [32], in which the authors hypothesize that the degrees of freedom of the sparse 1d fused lasso fit is equal to the number of nonzero fused coordinate groups, in expectation. But Corollary 2 covers any underlying graph, which makes it a much more general result.

By examining the null space of $D_{-\mathcal{B}}$ for other applications, and applying Theorem 2, one can obtain more corollaries on degrees of freedom. We omit the details for the sake of brevity, but list some such results in Table 2, along with those on the fused lasso for the sake of completeness. The table's first result, on the degrees of freedom of the lasso, was already established in [35]. The results on trend filtering and outlier detection can actually be derived from this lasso result, because these problems correspond to the case $\mathrm{rank}(D) = m$, and can be transformed into a regular lasso problem (11). For the outlier detection problem, we actually need to make a modification in order for the design matrix to have full column rank. Recall the problem formulation (8), where the coefficient vector is $(\alpha, \beta)^T$, the first block concerning the outliers, and the second the regression coefficients. We set $\alpha_1 = \ldots = \alpha_p = 0$, the interpretation being that we know a priori p points $y_1, \ldots y_p$ come from the true model, and only rest of the points $y_{p+1}, \ldots y_n$ can possibly be outliers (this is quite reasonable for a method that simultaneous performs a p-dimensional linear regression and detects outliers).

Problem	**Unbiased estimate of** $\mathrm{df}(X\hat{\beta}_\lambda)$
Lasso	Number of nonzero coordinates
Fused lasso	Number of fused groups
Sparse fused lasso	Number of nonzero fused groups
Polynomial trend filtering, order k	Number of knots $+ k + 1$
Outlier detection	Number of outliers $+ p$

Table 2: *Corollaries of Theorem 2, giving unbiased estimates of* $\mathrm{df}(X\hat{\beta}_\lambda)$ *for various problems discussed in Section 2. These assume that* $\mathrm{rank}(X) = p$.

10.2 Model selection

Note that the estimates Table 2 are all easily computable from the solution vector $\hat{\beta}_\lambda$. The estimates for the lasso, (sparse) fused lasso, and outlier detection problems can be obtained by simply counting the appropriate quantities in $\hat{\beta}_\lambda$. The estimate for trend filtering may be difficult to determine visually, as it may be difficult to identify the knots in a piecewise polynomial by eye, but the knots can be counted from the nonzeros of $D\hat{\beta}_\lambda$. All of this is important because it means that we can readily use model selection criteria like C_p or BIC for these problems, which employ degrees of freedom to assess risk. For example, for the estimate $X\hat{\beta}_\lambda$ of the underlying mean μ, the C_p statistic is

$$C_p(\lambda) = \|y - X\hat{\beta}_\lambda\|_2^2 - n\sigma^2 + 2\sigma^2 \mathrm{df}(X\hat{\beta}_\lambda),$$

(see [22] for the classic linear regression case), and is an unbiased estimate of the true risk $\mathrm{E}\big[\|\mu - X\hat{\beta}_\lambda\|_2^2\big]$. Hence we can define

$$\widehat{C}_p(\lambda) = \|y - X\hat{\beta}_\lambda\|_2^2 - n\sigma^2 + 2\sigma^2 \mathrm{nullity}(D_{-\mathcal{B}}),$$

replacing $\mathrm{df}(X\hat{\beta}_\lambda)$ by its own unbiased estimate $\mathrm{nullity}(D_{-\mathcal{B}})$. This modified statistic $\widehat{C}_p(\lambda)$ is still unbiased as an estimate of the true risk, and this suggests choosing λ to minimize $\widehat{C}_p(\lambda)$. For this task, it turns out that $\widehat{C}_p(\lambda)$ obtains its minimum at one of the critical points $\{\lambda_1, \ldots \lambda_T\}$ in the solution path of $\hat{\beta}_\lambda$. This is true because nullity$(D_{-\mathcal{B}})$ is a step function over these critical points, and the residual sum of squares $\|y - X\hat{\beta}_\lambda\|_2^2$ is monotone nondecreasing for λ in between critical points (this can be checked using (40)). Therefore Algorithm 2 can be used to simultaneously compute the solution path and select a model, by simply computing $\widehat{C}_p(\lambda_k)$ at each iteration k.

10.3 Shrinkage and the ℓ_1 norm

At first glance, the results in Table 2 seem both intuitive and unbelievable. For the fused lasso, for example, we are told that on average we spend a single degree of freedom on each group of coordinates in the solution. But these groups are being adaptively selected based on the data, so aren't we using more degrees of freedom in the end? As another example, consider the trend filtering result: for a cubic fit, the degrees of freedom is the number of knots + 4, in expectation. A cubic regression spline also has degrees of freedom equal to the number of knots + 4; however, in this case we fix the knot locations ahead of time, and for cubic trend filtering the knots are selected automatically. How can this be?

This seemingly remarkable property—that searching for the nonzero coordinates, fused groups, knots, or outliers doesn't cost us anything in terms of degrees of freedom—is explained by the shrinking nature of the ℓ_1 penalty. Looking back at the criterion in (2), it is not hard to see that the nonzero entries in $D\hat{\beta}_\lambda$ are shrunken towards zero (imagine the problem in constrained form, instead of Lagrangian form). For the fused lasso, this means that once the groups are "chosen", their coefficients are shrunken towards each other, which is less greedy than simply fitting the group coefficients to minimize the squared error term. Roughly speaking, this makes up for the fact that we chose the fused groups adaptively, and in expectation, the degrees of freedom turns out "just right": it is simply the number of groups.

This leads us to think about the ℓ_0-equivalent of problem (2), which is achieved by replacing the ℓ_1 norm by an ℓ_0 norm (giving best subset selection when $D = I$). Solving this problem requires a combinatorial optimization, and this makes it difficult to study the properties of its solution in general. However, we do know that the solution of the ℓ_0 problem does not enjoy any shrinkage property like that of the lasso solution: if we fix which entries of $D\beta$ are nonzero, then the penalty term is constant and the problem reduces to an equality-constrained regression. Therefore, in light of our above discussion, it seems reasonable to conjecture that the ℓ_0 fit has greater than E[nullity$(D_{-\mathcal{B}})$] degrees of freedom. When $D = I$, this would mean that the degrees of freedom of the best subset selection fit is more than the number of nonzero coefficients, in expectation.

11 Discussion

We have studied a generalization of the lasso problem, in which the penalty is $\|D\beta\|_1$ for a matrix D. Several important problems (such as the fused lasso and trend filtering) can be expressed as a special case of this, corresponding to a particular choice of D. We developed an algorithm to compute a solution path for this general problem, provided that the design matrix X has full column rank. This is achieved by instead solving the (easier) Lagrange dual problem, which, using simple duality theory, yields a solution to the original problem after a linear transformation.

Both the dual solution path and the primal (original) solution path are continuous and piecewise linear with respect to λ. The primal solution $\hat{\beta}_\lambda$ can be written explicitly in terms of the boundary set $\mathcal{B}$, which contains the coordinates of the dual solution that are equal to $\pm\lambda$, and the signs of these coordinates s. Further, viewing the dual solution as a projection onto a convex set, we derived a simple formula for the degrees of freedom of the generalized lasso fit. For the fused lasso problem, this result reveals that the number of nonzero fused groups in the solution is an unbiased estimate of the degrees of freedom of the fit, and this holds true for any underlying graph structure. Other corollaries follow, as well.

An implementation of our path algorithm, following the ideas presented in Section 8, is a direction for future work, and will be made available as an R package "genlasso" on the CRAN website [24].

There are several other directions for future research. We describe three possibilities below.

- *Specialized implementation for the fused lasso path algorithm.* When D is the fused lasso matrix corresponding to a graph $\mathcal{G}$, projecting onto the null space of $D_{-\mathcal{B}}$ is achieved by a simple coordinate-wise average on each connected component of $\mathcal{G}_{-\mathcal{B}}$. It may therefore be possible (when $X = I$) to compute the solution path $\hat{\beta}_\lambda$ without having to use any linear algebra, but by instead tracking the connectivity of $\mathcal{G}$. This could improve the computational efficiency of each iteration, and could also lead to a parallelized approach (in which we work on each connected component in parallel).

- *Number of steps until termination.* The number of steps T taken by our path algorithm, for a general D, is determined by how many times dual coordinates leave the boundary. This is related to an interesting problem in geometry studied by [10], and investigating this connection could lead to a more definitive statement about the algorithm's computational complexity.

- *Connection to forward stagewise regression.* When $D = I$ (and $\text{rank}(X) = p$), we proved that our path algorithm yields the LARS path (when LARS is run in its original, unmodified state) if we just ignore dual coordinates leaving the boundary. LARS can be modified to give forward stagewise regression, which is the limit of forward stepwise regression when the step size goes to zero (see [11]). A natural follow-up question is: can our algorithm be changed to give this path too?

In general, we believe that Lagrange duality deserves more attention in the study of convex optimization problems in statistics. The dual problem can have a complementary (and interpretable) structure, and can offer novel mathematical or statistical insights into the original problem.

Acknowledgements

The authors thank Robert Tibshirani for his many interesting suggestions and great support. Nick Henderson and Michael Saunders provided valuable input with the computational considerations. We also thank thank Trevor Hastie for his help with the LARS algorithm. Finally, we thank the referees and especially the Editor for all of their help making this paper more readable.

References

[1] Becker, S., Bobin, J. and Candes, E. J. [2011], 'NESTA: A fast and accurate first-order method for sparse recovery', *SIAM Journal on Imaging Sciences* **4**(1), 1–39.

[2] Bertsekas, D. P. [1999], *Nonlinear programming*, Athena Scientific, Nashua.

[3] Best, M. J. [1982], An algorithm for the solution of the parametric quadratic programming problem. CORR Report 82–84, University of Waterloo.

[4] Boyd, S. and Vandenberghe, L. [2004], *Convex Optimization*, Cambridge University Press, Cambridge.

[5] Bredel, M., Bredel, C., Juric, D., Harsh, G., Vogel, H., Recht, L. and Sikic, B. [2005], 'High-resolution genome-wide mapping of genetic alterations in human glial brain tumors', *Cancer Research* **65**(10), 4088–4096.

[6] Centers for Disease Control and Prevention [2009], 'Novel H1N1 flu situation update'.
URL: *http://www.cdc.gov/h1n1flu/updates/061909.htm*

[7] Chen, S., Donoho, D. and Saunders, M. [1998], 'Atomic decomposition for basis pursuit', *SIAM Journal on Scientific Computing* **20**(1), 33–61.

[8] Cleveland, W., Grosse, E., Shyu, W. and Terpenning, I. [1991], Local regression models, *in* J. Chambers and T. Hastie, eds, 'Statistical models in S', Wadsworth, Belmont.

[9] Donoho, D. and Johnstone, I. [1995], 'Adapting to unknown smoothness via wavelet shrinkage', *Journal of the American Statistical Association* **90**(432), 1200–1224.

[10] Donoho, D. L. and Tanner, J. [2010], 'Counting faces of randomly-projected hypercubes and orthants, with applications', *Dicrete and Computational Geometry* **43**(3), 522–541.

[11] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. [2004], 'Least angle regression', *Annals of Statistics* **32**(2), 407–499.

[12] Elad, M., Milanfar, P. and Rubinstein, R. [2007], 'Analysis versus synthesis in signal priors', *Inverse problems* **23**(3), 947–968.

[13] Evans, L. and Gariepy, R. [1992], *Measure theory and fine properties of functions*, CRC Press, Boca Raton.

[14] Friedman, J., Hastie, T., Hoefling, H. and Tibshirani, R. [2007], 'Pathwise coordinate optimization', *Annals of Applied Statistics* **1**(2), 302–332.

[15] Golub, G. and Loan, C. V. [1996], *Matrix computations*, The Johns Hopkins University Press, Baltimore. Third edition.

[16] Hastie, T., Rosset, S., Tibshirani, R. and Zhu, J. [2004], 'The entire regularization path for the support vector machine', *Journal of Machine Learning Research* **5**, 1391–1415.

[17] Hastie, T. and Tibshirani, R. [1990], *Generalized additive models*, Chapman and Hall, London.

[18] Hastie, T. and Tibshirani, R. [1993], 'Varying-coefficient models', *Journal of the Royal Statistical Society: Series B* **55**(4).

[19] Hoefling, H. [2009], A path algorithm for the fused lasso signal approximator. Unpublished.
URL: *http://www.holgerhoefling.com/Articles/FusedLasso.pdf*

[20] James, G. M., Radchenko, P. and Lv, J. [2009], 'DASSO: connections between the Dantzig selector and lasso', *Journal of the Royal Statistical Society: Series B* **71**(1), 127–142.

[21] Kim, S.-J., Koh, K., Boyd, S. and Gorinevsky, D. [2009], 'ℓ_1 trend filtering', *SIAM Review* **51**(2), 339–360.

[22] Mallows, C. [1973], 'Some comments on C_p', *Technometrics* **15**(4), 661–675.

[23] Osborne, M., Presnell, B. and Turlach, B. [2000], 'On the lasso and its dual', *Journal of Computational and Graphical Statistics* **9**(2), 319–337.

[24] R Development Core Team [2008], *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL: *http://www.R-project.org*

[25] Rosset, S. and Zhu, J. [2007], 'Piecewise linear regularized solution paths', *Annals of Statistics* **35**(3), 1012–1030.

[26] Rudin, L. I., Osher, S. and Faterni, E. [1992], 'Nonlinear total variation based noise removal algorithms', *Physica D: Nonlinear Phenomena* **60**, 259–268.

[27] Schneider, R. [1993], *Convex bodies: the Brunn-Minkowski theory*, Cambridge University Press, Cambridge.

[28] She, Y. [2010], 'Sparse regression with exact clustering', *Electronic Journal of Statistics* **4**, 1055–1096.

[29] She, Y. and Owen, A. B. [2011], 'Outlier detection using nonconvex penalized regression', *Journal of the American Statistical Association: Theory and Methods* **106**(494), 626–639.

[30] Stein, C. [1981], 'Estimation of the mean of a multivariate normal distribution', *Annals of Statistics* **9**(6), 1135–1151.

[31] Tibshirani, R. [1996], 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society: Series B* **58**(1), 267–288.

[32] Tibshirani, R., Saunders, M., Rosset, S., Zhu, J. and Knight, K. [2005], 'Sparsity and smoothness via the fused lasso', *Journal of the Royal Statistics Society: Series B* **67**(1), 91–108.

[33] Tseng, P. [2001], 'Convergence of a block coordinate descent method for nondifferentiable minimization', *Journal of Optimization Theory and Applications* **109**(3), 475–494.

[34] Zou, H. and Hastie, T. [2005], 'Regularization and variable selection via the elastic net', *Journal of the Royal Statistical Society: Series B* **67**(2), 301–320.

[35] Zou, H., Hastie, T. and Tibshirani, R. [2007], 'On the "degrees of freedom" of the lasso', *Annals of Statistics* **35**(5), 2173–2192.