Tuning Parameter Selection based on Validation-Error Descent

Abstract

In high-dimensional and/or non-parametric regression problems, regularization (or penalization) is used to control model complexity and induce desired structure. Each penalty has a weight parameter that indicates how strongly the structure corresponding to that penalty should be enforced. To date, for problems with $k = 2$ or more penalties, tuning these penalty parameters is a challenge. The current gold-standard of calculating validation error over a k-dimensional grid of parameter values quickly becomes computationally intractable as k increases. We propose tuning parameters by solving a continuous optimization problem over a validation set and updating the values using a descent-based approach. We show that our method is significantly more efficient than calculating validation error over an entire grid, and empirically achieves the same performance (on scenarios where a grid search could be performed). This descent-based approach enables us to test regularization problems with many penalty parameters, through which we discover new regularization methods with superior accuracy. We also include simulated experiments, and a data analysis, which illustrate the strength of this new method.

Keywords: regularization, high-dimensional regression, cross-validation, optimization

1 Introduction

Consider the usual regression framework with p features, $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^\top$, and a response y_i measured on each of $i = 1, \ldots, n$ observations. Let $\boldsymbol{X}$ denote the $n \times p$ design matrix and $\boldsymbol{y}$ the response vector. Our goal here is to characterize the conditional relationship between $\boldsymbol{y}$ and $\boldsymbol{X}$. In simple low-dimensional problems this is often done by constructing an f in some pre-specified class $\mathcal{F}$ that minimizes a measure of discrepancy between $\boldsymbol{y}$ and $f(\boldsymbol{X})$. Generally, this discrepancy is quantified with some pre-specified loss, L. Often $\mathcal{F}$ will endow f with some simple form (e.g. a linear function). For ill-posed or high-dimensional problems ($p \gg n$), there can often be an infinite number of solutions that minimize the loss function L but have high generalization error. A common solution is to use regularization, or penalization, to select models with desirable properties, such as smoothness and sparsity.

In recent years, there has been much interest in combining regularization methods to produce models with multiple desired characteristics. Examples include the elastic net (Zou and Hastie 2003), which combines the lasso and ridge penalties, and the sparse group lasso (Simon et al. 2013), which combines the group lasso and lasso penalties. The general form of these regression problems is:

$$\hat{f}(\boldsymbol{\lambda}) = \underset{f \in \mathcal{F}}{\arg\min}\, L\left(\boldsymbol{y}, f(\boldsymbol{X})\right) + \sum_{i=1}^{J} \lambda_i P_i(f) \tag{1}$$

where $\{P_i\}_{i=1,\ldots,J}$ are the penalty functions and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_J)^\top$ are the regularization parameters.

Regularization parameters control the degree of various facets of model complexity, such as the amount of sparsity or smoothness. Often, the goal is to set the parameters to minimize the fitted model's generalization error. One usually estimates this using a training/validation approach (or cross validation). There one fits a model on a training set $(\boldsymbol{X}_T, \boldsymbol{y}_T)$ and measures the model's error on a validation set $(\boldsymbol{X}_V, \boldsymbol{y}_V)$. The goal then is to choose penalty parameters $\boldsymbol{\lambda}$ that minimize the validation error, as formulated in the following optimization problem:

$$\begin{aligned} &\min\nolimits_{\boldsymbol{\lambda} \in \Lambda} L\left(\boldsymbol{y}_V, \hat{f}(\boldsymbol{X}_V | \boldsymbol{\lambda})\right) \\ \text{s.t. } &\hat{f}(\cdot|\boldsymbol{\lambda}) = \arg\min\nolimits_{f \in \mathcal{F}} L\left(\boldsymbol{y}_T, f(\boldsymbol{X}_T)\right) + \sum_{i=1}^{J} \lambda_i P_i(f) \end{aligned} \tag{2}$$

Here Λ is some set that $\boldsymbol{\lambda}$ are known to be in, which is often just $\mathbb{R}_+^J$.

The simplest approach to solving (2) is brute force: one fits models over a grid of parameter values and selects the model with the lowest validation error. As long as the grid is large and fine enough, this method of "grid search" will find a solution close to the global optimum. This approach is the current standard for choosing penalty parameters via training/validation. Unfortunately, it is computationally intractable in cases with more than two parameters since the runtime is exponential in the number of parameters. For certain special cases, there are more efficient ways of tuning the parameters (Golub et al. 1979), (Wood 2000), but there is no general solution to date.

In this paper, we propose leveraging the tools of optimization to solve (2). We give a gradient descent algorithm for minimizing the validation error over the penalty parameter space. In contrast to an exhaustive "grid search", this "descent-based" optimization makes use of the smoothness of our validation-error surface. (2) is generally not convex and thus we may not find the global minimum with a simple descent-based approach. However, in practice we find that simple descent gives competitive solutions.

In simulation studies we show that our descent-based optimization produces solutions with the same validation error as those from grid search. In addition, we find that our approach is highly efficient and can solve regressions with hundreds of penalty parameters. Finally, we use this method to analyze regularization methods that were previously computationally intractable. Through this, we discover that a variant of sparse group lasso with many more penalty parameters can significantly decrease error and produce more meaningful models.

Lorbert and Ramadge (2010) presented some related work on this topic. They solved linear regression problems by updating regression coefficients and regularization parameters using cyclical coordinate gradient descent. We take a more general approach that allows us to apply this descent-based optimization to a wide array of problems. We present examples in this paper that demonstrate the wide applicability of our method.

In Section 2, we describe descent-based optimization in detail and present an algorithm for solving it in example regressions. In Section 3, we show that our method achieves validation errors as low as those achieved by grid search. In Section 3, we explore variants of the example regression problems that have many more regularization parameters and demonstrate that solving (2) is still computationally tractable. Finally, we present results on data predicting colitis status from gene expression in Section 5.

2 Descent-based Joint Optimization

2.1 Definition

In this manuscript we will restrict ourselves to classes $\mathcal{F} = \{f_{\boldsymbol{\theta}} | \boldsymbol{\theta} \in \Theta\}$, which, for a fixed sample size n, are in some finite dimensional space Θ. This is not a large restriction: the class of linear functions functions meets this requirement; as does any class of finite dimensional parametric functions. Even non-parametric methods generally either use a growing basis expansion (e.g. Polynomial regression, smoothing-splines, wavelet-based-regression, locally-adaptive regression splines (Tsybakov 2008), (Wahba 1981), (Donoho and Johnstone 1994), (Mammen et al. 1997)), or only evaluate the function at the observed data-points (eg. trend filtering, fused lasso, (Kim et al. 2009), (Tibshirani et al. 2005)). In these non-parametric problems, for any fixed n, $\mathcal{F}$ is representable as a finite dimensional class. We can therefore rewrite (1) in the following form:

$$\underset{\boldsymbol{\theta} \in \Theta}{\arg\min}\, L(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{X})) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{\theta}) \tag{3}$$

Suppose that we use a training/validation split to select penalty parameters $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_J)^\top$. Let the data be partitioned into a training set $(\boldsymbol{y}_T, \boldsymbol{X}_T)$ and validation set $(\boldsymbol{y}_V, \boldsymbol{X}_V)$. We can rewrite the joint optimization problem (2) over this finite-dimensional class as:

$$\begin{array}{c} \arg\min_{\boldsymbol{\lambda} \in \Lambda} L(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ \text{s.t. } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta} \in \Theta} L(\boldsymbol{y}_T, f_{\boldsymbol{\theta}}(\boldsymbol{X}_T)) + \sum\limits_{i=1}^{J} \lambda_i P_i(\boldsymbol{\theta}) \end{array} \tag{4}$$

For the remainder of the manuscript we will assume that (3) for the training set is strictly convex in $\boldsymbol{\theta}$. This ensures that there is a unique $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ which perturbs continuously in $\boldsymbol{\lambda}$.

(4) is the explicit, though often unstated, criterion that training/validation methods attempt to minimize when choosing penalty parameters. The current standard is to minimize this using an exhaustive grid search. Grid-based methods solve the joint optimization

problem by fitting models over a J-dimensional grid in the penalty parameter space; so the computational runtime grows exponentially with the number of penalty parameters. While the approach is simple and powerful for a single penalty parameter, optimizing even moderate-dimensional functions (3+) via exhaustive grid search is inefficient (and quickly becomes completely intractable). In addition, (4) is generally a continuous, piecewise-smooth problem. An exhaustive search ignores information available from the smoothness of the surface.

We propose solving (4) by using the tools of smooth optimization. In particular we discuss iterative methods, based on walking in a descent direction until convergence to a local minimum. In the simple case where the criterion is differentiable with respect to the penalty parameters, it is straightforward to use gradient descent or some variant thereof. We show that, with some slight tweaks, gradient descent can be also applied in situations where the penalty is only differentiable in certain directions.

Figure 1 illustrates the difference between these two approaches. Grid search fits a model at every grid point, many of which are not close to the global (or local) minima. In contrast, descent-based methods incorporate information about the shape of the local neighborhood to choose an intelligent descent direction. It explores the space more efficiently since it avoids penalty parameter values unlikely to yield good models.

To ease exposition, we will assume throughout the remainder of the manuscript that $L\left(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V)\right)$ is differentiable in $\boldsymbol{\theta}$. This assumption is met if both 1) $f_{\boldsymbol{\theta}}(\boldsymbol{X}_V)$ is continuous as a function of $\boldsymbol{\theta}$; and 2) $L\left(\boldsymbol{y}_V, \cdot\right)$ is smooth. Examples include the squared-error, logistic, and poisson loss functions, though not the hinge loss.

2.2 Smooth Training Criterion

Let us denote the training criterion as follows

$$L_T\left(\boldsymbol{\theta}, \boldsymbol{\lambda}\right) \equiv L(\boldsymbol{y}_T, f_{\boldsymbol{\theta}}(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{\theta}) \tag{5}$$

First we consider the simple case where $L_T\left(\boldsymbol{\theta}, \boldsymbol{\lambda}\right)$ is smooth as a function of $(\boldsymbol{\theta}, \boldsymbol{\lambda})$. In this case, the validation loss is differentiable as a function of $\boldsymbol{\lambda}$. So we can directly apply gradient descent to solve (4), as described in Algorithm 1.

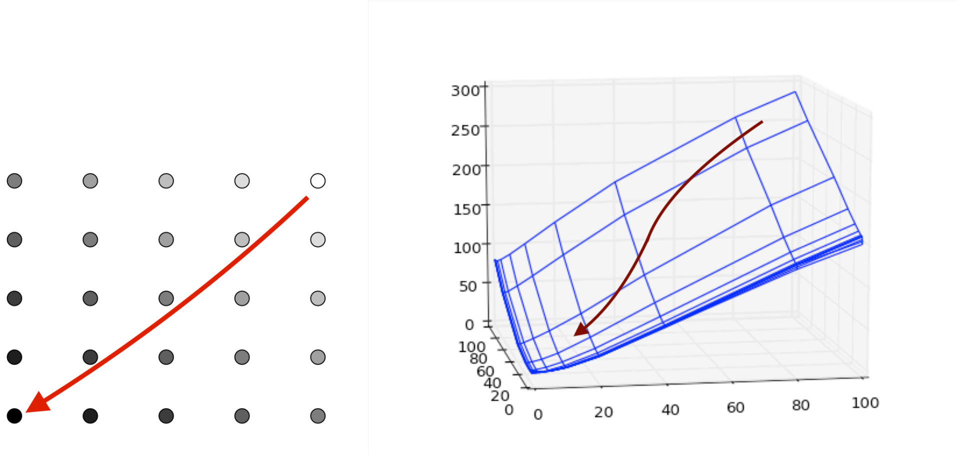


Figure 1: Left: A hypothetical grid of (λ_1, λ_2) points that an exhaustive grid search would fit models for. The darkness of each point indicates the validation cost; dark points mean lower cost. In this example, descent-based optimization would takes steps along the arrow, while a grid search would have to consider all 25 points, many of which are obviously poor candidates. Right: The same example with validation loss now on the vertical axis.

Algorithm 1 Gradient Descent for Smooth Training Criterions

```
Initialize $\boldsymbol{\lambda}^{(0)}$.
**for** each iteration $k = 0, 1, ...$ until stopping criteria is reached **do**
    Perform gradient step with step size $t^{(k)}$
        $$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - t^{(k)} \nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)\Big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} \quad (6)$$
**end for**
```

There are a number of potential ways to choose the step-size $t^{(k)}$ — two simple options are fixed size $t^{(k)} = t$ and harmonically decreasing $t^{(k)} = t/k$. Choice of step-size is discussed further in Section 2.6.

Calculating the Gradient: The gradient can be found using the chain rule:

$$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right) = \left[\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V)) \Big|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})} \right]^\top \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \tag{7}$$

The first term, $\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V))$, is problem specific, but generally straightforward to calculate. To calculate the second term, $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$, we note that $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ minimizes (5). Since (5) is smooth,

$$\nabla_\theta \left(L(\boldsymbol{y}_T, f_{\boldsymbol{\theta}}(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{\theta}) \right) \Bigg|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})} = \boldsymbol{0}. \tag{8}$$

Taking the derivative of both sides of (8) in $\boldsymbol{\lambda}$ and solving for $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$, we get:

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = - \left[\left[\nabla_\theta^2 \left(L\left(\boldsymbol{y}_T, f_{\boldsymbol{\theta}}(\boldsymbol{X}_T)\right) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{\theta}) \right) \right]^{-1} \nabla_{\boldsymbol{\theta}} P(\boldsymbol{\theta}) \right] \Bigg|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})} \tag{9}$$

where $\nabla_{\boldsymbol{\theta}} P(\boldsymbol{\theta})$ is the matrix with columns $\{\nabla_{\boldsymbol{\theta}} P_i(\boldsymbol{\theta})\}_{i=1:J}$.

We can plug (9) into (7) to get $\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)$. Note that because $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is defined in terms of $\hat{\boldsymbol{\theta}}\left(\boldsymbol{\lambda}\right)$, each gradient step requires minimizing the training criterion first. Algorithm 2 is the updated version of Algorithm 1 with the specific gradient calculations.

2.3 Nonsmooth Training Criterion

When the penalized training criterion in the joint optimization problem is not smooth, gradient descent cannot be applied. Nonetheless, we find that in many problems, the solution $\hat{\boldsymbol{\theta}}\left(\boldsymbol{\lambda}\right)$ is smooth at almost every $\boldsymbol{\lambda}$ (eg. Lasso, Group Lasso, Trend Filtering); this means that we can indeed apply gradient descent in practice. In this section, we characterize these problems that are almost everywhere smooth. In addition, we provide a solution for deriving $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ since calculating the gradient is a challenge in and of itself. This is then incorporated into an algorithm for tuning $\boldsymbol{\lambda}$ using gradient descent.

Algorithm 2 Updated Algorithm 1

```
Initialize $\boldsymbol{\lambda}^{(0)}$.
for each iteration $k = 0, 1, ...$ until stopping criteria is reached do
    Solve for $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k)}) = \arg\min_{\boldsymbol{\theta}\in\Theta} L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(k)})$.
    Calculate the derivative of the model parameters with respect to the regularization parameters
        $$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = - \left[ \left[ \nabla_\theta^2 \left( L\left(\boldsymbol{y_T}, f_\theta(\boldsymbol{X}_T)\right) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{\theta}) \right) \right]^{-1} \nabla_{\boldsymbol{\theta}} P(\boldsymbol{\theta}) \right] \Bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k)})} \tag{10}$$
    Calculate the gradient
        $$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X_V})\right)\Big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} = \left[\frac{\partial}{\partial\boldsymbol{\theta}} L(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X_V}))\Big|_{\theta=\hat{\theta}(\boldsymbol{\lambda}^{(k)})}\right]^\top \frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} \tag{11}$$
    Perform gradient step with step size $t^{(k)}$
        $$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - t^{(k)} \nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)\Big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} \tag{12}$$
end for
```

To characterize problems that are almost everywhere smooth, we begin with three definitions:

Definition 1. *The differentiable space of a real-valued function L at a point $\boldsymbol{\eta}$ in its domain is the set of vectors along which the directional derivative of L exists.*

$$\Omega^{L}(\boldsymbol{\eta}) = \left\{ \boldsymbol{u} \middle| \lim_{\epsilon \to 0} \frac{L(\boldsymbol{\eta} + \epsilon \boldsymbol{u}) - L(\boldsymbol{\eta})}{\epsilon} \text{ exists} \right\} \tag{13}$$

Definition 2. *S is a local optimality space for a convex function $L(\cdot, \boldsymbol{\lambda}_0)$ if there exists a neighborhood W containing $\boldsymbol{\lambda}_0$ such that for every $\boldsymbol{\lambda} \in W$,*

$$\arg\min_{\boldsymbol{\theta} \in \Theta} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta} \in S} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{14}$$

Definition 3. *Let matrix $\boldsymbol{B} = [\boldsymbol{b}_1 \dots \boldsymbol{b}_p] \in \mathbb{R}^{n \times p}$ have orthonormal columns. Let f be a real-valued function over $\mathbb{R}^n$ and suppose its first and second directional derivatives of f with respect to the columns in $\boldsymbol{B}$ exist. The Gradient vector and Hessian matrix of f with respect to $\boldsymbol{B}$ are defined respectively as*

$$_{\boldsymbol{B}}\nabla f \in \mathbb{R}^p = \begin{pmatrix} \frac{\partial f}{\partial \boldsymbol{b}_1} \\ \frac{\partial f}{\partial \boldsymbol{b}_2} \\ \vdots \\ \frac{\partial f}{\partial \boldsymbol{b}_p} \end{pmatrix}; \quad _{\boldsymbol{B}}\nabla^2 f \in \mathbb{R}^{p \times p} = \begin{pmatrix} \frac{\partial^2 f}{\partial b_1^2} & \frac{\partial^2 f}{\partial b_1 \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_1 \partial b_p} \\ \frac{\partial^2 f}{\partial b_2 \partial b_1} & \frac{\partial^2 f}{\partial b_2^2} & \cdots & \frac{\partial^2 f}{\partial b_2 \partial b_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial b_p \partial b_1} & \frac{\partial^2 f}{\partial b_p \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_p^2} \end{pmatrix} \tag{15}$$

Using these definitions we can now give three conditions which together are sufficient for the differentiability of $L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)$ almost everywhere.

Condition 1. *For almost every $\boldsymbol{\lambda}$, the differentiable space $\Omega^{L_T(\cdot, \boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$ is a local optimality space for $L_T(\cdot, \boldsymbol{\lambda})$.*

Condition 2. *For almost every $\boldsymbol{\lambda}$, $L_T(\cdot, \cdot)$ restricted to $\Omega^{L_T(\cdot,\cdot)}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}), \boldsymbol{\lambda})$ is twice continuously differentiable within some neighborhood of $\boldsymbol{\lambda}$.*

Condition 3. *For almost every $\boldsymbol{\lambda}$, there exists an orthonormal basis $\boldsymbol{B}$ of $\Omega^{L_T(\cdot, \boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$ such that the Hessian of $L_T(\cdot, \boldsymbol{\lambda})$ at $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ with respect to $\boldsymbol{B}$ is invertible.*

Note that if condition 3 is satisfied, the Hessian of $L_T(\cdot, \boldsymbol{\lambda})$ with respect to any orthonormal basis of $\Omega^{L_T(\cdot, \boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$ is invertible.

Putting all these conditions together, the following theorem establishes that the gradient exists almost everywhere and provides a recipe for calculating it.

Theorem 1. *Suppose our optimization problem is of the form in* (4)*, with* $L_T\left(\boldsymbol{\theta}, \boldsymbol{\lambda}\right)$ *defined as in* (5)*.*

Suppose that $L\Big(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V)\Big)$ *is continuously differentiable in* $\boldsymbol{\theta}$*, and conditions* 1, 2*, and* 3*, defined above, hold.*

Then the validation loss $L(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_{\boldsymbol{V}}))$ *is continuously differentiable with respect to* $\boldsymbol{\lambda}$ *for almost every* $\boldsymbol{\lambda}$*. Furthermore, the gradient of* $L(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_{\boldsymbol{V}}))$*, where it is defined, is*

$$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_{\boldsymbol{V}})\right) = \left[\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{y}_{\boldsymbol{V}}, f_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{V}}))\Big|_{\boldsymbol{\theta}=\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})}\right]^\top \frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \tag{16}$$

where

$$\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \underset{\boldsymbol{\theta} \in \Omega^{L_T(\cdot, \boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))}{\arg\min} \; L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{17}$$

We can therefore construct a gradient descent procedure based on the model parameter constraint in (17). At each iteration, let matrix $\boldsymbol{U}$ have orthonormal columns spanning the differentiable space $\Omega^{L_T(\cdot, \boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$. Since this space is also a local optimality space, it is sufficient to minimize the training criterion over the column space of $\boldsymbol{U}$. The joint optimization problem can be reformulated using $\boldsymbol{U}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ as the model parameters instead:

$$\begin{gathered} \min_{\boldsymbol{\lambda} \in \Lambda} L(\boldsymbol{y}_V, f_{\boldsymbol{U}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ \text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} L(\boldsymbol{y}_T, f_{\boldsymbol{U}\boldsymbol{\beta}}(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{U}\boldsymbol{\beta}) \end{gathered} \tag{18}$$

This locally equivalent problem now reduces to the simple case where the training criterion is smooth. As mentioned previously, implicit differentiation on the gradient condition then gives us $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$, which gives us the value of interest

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \boldsymbol{U} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \tag{19}$$

Note that because the differentiable space is a local optimality space and is thus locally constant, we can treat $\boldsymbol{U}$ as a constant in the gradient derivations. Algorithm 3 provides the exact steps for tuning the regularization parameters.

Thus far, we have restricted our attention to joint optimization for training/validation splits. We can also perform joint optimization for K-fold cross validation by reformulating

Algorithm 3 Joint Optimization with Gradient Descent

```
Initialize $\boldsymbol{\lambda}^{(0)}$.
for each iteration $k = 0, 1, ...$ until stopping criteria is reached do
    Solve for $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k)}) = \arg\min_{\theta \in \Theta} L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(k)})$.
    Construct matrix $\boldsymbol{U}^{(k)}$, an orthonormal basis of $\Omega^{L_T(\cdot, \boldsymbol{\lambda})}\left(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k)})\right)$.
    Define the locally equivalent joint optimization problem
        $$\begin{gathered} \min_{\boldsymbol{\lambda} \in \Lambda} L(\boldsymbol{y}_V, f_{\boldsymbol{U}^{(k)}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ \text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} L(\boldsymbol{y}_T, f_{\boldsymbol{U}^{(k)}\boldsymbol{\beta}}(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{U}^{(k)}\boldsymbol{\beta}) \end{gathered} \tag{20}$$
    Calculate $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(k)}}$ where
        $$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\left[ {}_{\boldsymbol{U}^{(k)}}\nabla^2 \left( L(\boldsymbol{y}_T, f_{\boldsymbol{U}^{(k)}\boldsymbol{\beta}}(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{U}^{(k)}\boldsymbol{\beta}) \right) \Bigg|_{\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right]^{-1} {}_{\boldsymbol{U}^{(k)}}\nabla P(\boldsymbol{U}^{(k)}\boldsymbol{\beta}) \Big|_{\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \tag{21}$$
    with ${}_{\boldsymbol{U}^{(k)}}\nabla^2$ and ${}_{\boldsymbol{U}^{(k)}}\nabla$ are as defined in (15).
    Calculate the gradient $\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_{\boldsymbol{V}}))|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(k)}}$ where
        $$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_{\boldsymbol{V}})\right) = \left[\boldsymbol{U}^{(k)} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right]^{\top} \left[ {}_{\boldsymbol{U}^{(k)}}\nabla L\left(\boldsymbol{y}_{\boldsymbol{V}}, f_{\boldsymbol{U}^{(k)}\boldsymbol{\beta}}(\boldsymbol{X}_{\boldsymbol{V}})\right)\big|_{\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right] \tag{22}$$
    Perform the gradient update with step size $t^{(k)}$
        $$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - t^{(k)} \, \nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_{\boldsymbol{V}})\right)\Big|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(k)}}$$
end for
```

the problem. Let $(\boldsymbol{y}, \boldsymbol{X})$ be the full data set. We denote the kth fold as $(\boldsymbol{y}_k, \boldsymbol{X}_k)$ and its complement as $(\boldsymbol{y}_{-k}, \boldsymbol{X}_{-k})$. Then the objective of this joint optimization problem is the average validation cost across all K folds:

$$\begin{array}{c} \arg\min_{\boldsymbol{\lambda}\in\Lambda} \frac{1}{K}\sum_{k=1}^{K} L(\boldsymbol{y}_k, f_{\hat{\boldsymbol{\theta}}^{(k)}(\boldsymbol{\lambda})}(\boldsymbol{X}_k)) \\ \text{s.t. } \hat{\boldsymbol{\theta}}^{(k)}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}\in\Theta} L(\boldsymbol{y}_{-k}, f_{\boldsymbol{\theta}}(\boldsymbol{X}_{-k})) + \sum_{i=1}^{J} \lambda_i P_i(\boldsymbol{\theta}) \text{ for } k = 1, ..., K \end{array} \tag{23}$$

2.4 Examples

To better understand the proposed gradient descent procedure, we present example joint optimization problems and their corresponding gradient calculations. We start with ridge regression where the training criterion is smooth. Then we consider the elastic net, sparse group lasso, and the generalized lasso, where the training criterions are nonsmooth, but $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is smooth almost everywhere. Finally, we discuss descent-based joint optimization for an additive partially linear model as an example of semi-parametric regression and an additive model as an example of nonparametric regression.

For ease of notation, we will let $S_{\boldsymbol{\lambda}}$ denote the differentiable space of $L_T(\cdot, \boldsymbol{\lambda})$ at $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$. For reference, $S_{\boldsymbol{\lambda}}$ for each regression example is specified in Table 1.

Note that in some of the examples below, we add a ridge penalty with a fixed small coefficient $\epsilon > 0$ to ensure that the training criterion is strictly convex.

2.4.1 Ridge Regression

In ridge regression, the training criterion is smooth so applying gradient descent is straightforward. The joint optimization problem for ridge regression is:

$$\begin{array}{c} \min_{\lambda\in\mathbb{R}_+} \frac{1}{2}\|\boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\theta}}(\lambda)\|_2^2 \\ \text{where } \hat{\boldsymbol{\theta}}(\lambda) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\lambda\|\boldsymbol{\theta}\|_2^2 \end{array} \tag{24}$$

The closed-form solution for $\hat{\boldsymbol{\theta}}(\lambda)$ is

$$\hat{\boldsymbol{\theta}}(\lambda) = (\boldsymbol{X}_T^\top \boldsymbol{X}_T + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}_T^\top \boldsymbol{y}_T \tag{25}$$

The gradient of the validation loss can be easily derived by differentiating the above equation with respect to λ and then using the chain rule.

	Differentiable Space
Ridge Regression	$\mathbb{R}^p$
Elastic Net	$span(\{e_i \vert \hat{\theta}_i\,(\boldsymbol{\lambda}) \neq 0 \text{ for } i = 1, ..., p\})$
Sparse Group Lasso	$span(\{e_i \vert \hat{\theta}_i\,(\boldsymbol{\lambda}) \neq 0 \text{ for } i = 1, ..., p\})$
Generalized Lasso	$\mathcal{N}(\boldsymbol{I}_{I(\boldsymbol{\lambda})}\boldsymbol{D})$ where $I(\boldsymbol{\lambda}) = \{i \vert \left(\boldsymbol{D}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\right)_i = 0 \text{ for } i = 1, ..., p\}$
Additive Partially Linear Model	$span(\{e_i \vert \hat{\beta}_i\,(\boldsymbol{\lambda}) \neq 0 \text{ for } i = 1, ..., p\}) \oplus \mathbb{R}^n$
Additive Model	$\mathbb{R}^p$

Table 1: The differentiable space of example regression problems in Section 2.4

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X_V})) \;\; = \;\; (\boldsymbol{X}_V(\boldsymbol{X}_T^\top \boldsymbol{X}_T + \lambda \boldsymbol{I})^{-1}\hat{\boldsymbol{\theta}}(\lambda))^\top(\boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\theta}}(\lambda)) \tag{26}$$

2.4.2 Elastic Net

The elastic net (Zou and Hastie 2003), a linear combination of the lasso and ridge penalties, is an example of a regularization method that is not smooth. We are interested in choosing regularization parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^\top$ using the following joint optimization problem:

$$\begin{array}{c} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \frac{1}{2}\|\boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\|^2 \\ \text{s.t. } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\|^2 + \lambda_1\|\boldsymbol{\theta}\|_1 + \frac{1}{2}\lambda_2\|\boldsymbol{\theta}\|_2^2 \end{array} \tag{27}$$

Let the nonzero indices of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ be denoted $I(\boldsymbol{\lambda}) = \{i|\hat{\theta}_i(\boldsymbol{\lambda}) \neq 0 \text{ for } i = 1, ..., p\}$ and let $\boldsymbol{I}_{I(\boldsymbol{\lambda})}$ be a submatrix of the identity matrix with columns $I(\boldsymbol{\lambda})$. Since $|\cdot|$ is not differentiable at zero, the directional derivatives of $||\boldsymbol{\theta}||_1$ only exist along directions spanned by the columns of $\boldsymbol{I}_{I(\boldsymbol{\lambda})}$. That is, the differentiable space at $\boldsymbol{\lambda}$ is

$$S_{\boldsymbol{\lambda}} = span(\boldsymbol{I}_{I(\boldsymbol{\lambda})}) \tag{28}$$

Next, we show that the joint optimization problem satisfies all three conditions in Theorem 1:

Condition 1: The nonzero indices of the elastic net estimates stay locally constant for almost every $\boldsymbol{\lambda}$. Therefore, $S_{\boldsymbol{\lambda}}$ is a local optimality space for $L_T(\cdot, \boldsymbol{\lambda})$ ✓

Condition 2: The ℓ_1 penalty is smooth when restricted to $S_{\boldsymbol{\lambda}}$. ✓

Condition 3: The Hessian matrix of $L_T(\cdot, \boldsymbol{\lambda})$ with respect to the columns of $\boldsymbol{I}_{I(\boldsymbol{\lambda})}$ is $\boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} + \lambda_2 \boldsymbol{I}$. This is positive definite if $\lambda_2 > 0$. ✓

To calculate the gradient, we consider the locally equivalent joint optimization problem

$$\begin{array}{c} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \frac{1}{2} \|\boldsymbol{y}_V - \boldsymbol{X}_V \boldsymbol{I}_{I(\boldsymbol{\lambda})} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\|^2 \\ \text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} \boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{I}_{I(\boldsymbol{\lambda})} \boldsymbol{\beta}\|_1 + \frac{1}{2} \lambda_2 \|\boldsymbol{I}_{I(\boldsymbol{\lambda})} \boldsymbol{\beta}\|_2^2 \end{array} \tag{29}$$

This can be further simplified by defining $\boldsymbol{X}_{T,I(\boldsymbol{\lambda})} = \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})}$ and $\boldsymbol{X}_{V,I(\boldsymbol{\lambda})} = \boldsymbol{X}_V \boldsymbol{I}_{I(\boldsymbol{\lambda})}$, which are submatrices of $\boldsymbol{X}_T$ and $\boldsymbol{X}_V$ with columns $I(\boldsymbol{\lambda})$. The simplified optimization problem is

$$\begin{array}{c} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \frac{1}{2} \|\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\|^2 \\ \text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} \boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \frac{1}{2} \lambda_2 \|\boldsymbol{\beta}\|_2^2 \end{array} \tag{30}$$

Since the training criterion is now smooth, we can apply (9) to get

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \left(\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} + \lambda_2 \boldsymbol{I} \right)^{-1} \left[sgn\left(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right) \quad \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right] \tag{31}$$

Hence, the gradient descent direction at $\boldsymbol{\lambda}$ is

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\boldsymbol{\theta}}(\lambda)}(\boldsymbol{X}_{\boldsymbol{V}})) = \left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right) \tag{32}$$

2.4.3 Sparse Group Lasso

The sparse group lasso combines the $\|\cdot\|_2$ and $\|\cdot\|_1$ penalties, both of which are not smooth (Simon et al. 2013). This method is particularly well-suited for problems where features have a natural grouping, and only a few of the features from a few of the groups are thought to have an effect on response (e.g. genes in gene pathways).

The problem setup is as follows. Given M covariate groups, suppose $\boldsymbol{X}$ and $\boldsymbol{\theta}$ are partitioned into $\boldsymbol{X}^{(m)}$ and $\boldsymbol{\theta}^{(m)}$ for groups $m = 1, ..., M$. We are interested in finding the optimal regularization parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^\top$. The joint optimization problem is formulated as follows.

$$\begin{array}{c} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2_+} \frac{1}{2n}\left\|\boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\right\|_2^2 \\ \text{s.t. } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2n}\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\|_2^2 + \lambda_1\sum_{m=1}^M \|\boldsymbol{\theta}^{(m)}\|_2 + \lambda_2\|\boldsymbol{\theta}\|_1 + \frac{1}{2}\epsilon\|\boldsymbol{\theta}\|_2^2 \end{array} \tag{33}$$

Note the addition of a small, fixed ridge penalty to ensure strong convexity. As $\|\cdot\|_2$ (or $|\cdot|$) is not differentiable in any direction at $\mathbf{0}$ (or 0) and is differentiable in all directions elsewhere, it is straightforward to show that

$$S_{\boldsymbol{\lambda}} = span(\boldsymbol{I}_{I(\boldsymbol{\lambda})}) \tag{34}$$

where $I(\boldsymbol{\lambda}) = \{i|\hat{\theta}_i(\boldsymbol{\lambda}) \neq 0 \text{ for } i = 1, ..., p\}$ are the nonzero indices of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$.

This problem satisfies all three conditions in Theorem 1. Since the reasoning for the first two conditions is exactly the same, we just give the calculations for the third condition.

> Condition 3: The Hessian matrix of $L_T(\cdot, \boldsymbol{\lambda})$ with respect to the columns of $\boldsymbol{I}_{I(\boldsymbol{\lambda})}$ is
>
> $$\frac{1}{n}\boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} + \lambda_1 \boldsymbol{B}(\boldsymbol{\lambda}) + \epsilon \boldsymbol{I}_p \tag{35}$$
>
> where $\boldsymbol{B}(\boldsymbol{\lambda})$ is a block diagonal matrix with components
>
> $$\left\|\tilde{\boldsymbol{\theta}}^{(m)}\boldsymbol{\lambda})\right\|_2^{-1}\left(\boldsymbol{I} - \frac{\tilde{\boldsymbol{\theta}}^{(m)}(\boldsymbol{\lambda})\tilde{\boldsymbol{\theta}}^{(m)}(\boldsymbol{\lambda})^\top}{\|\tilde{\boldsymbol{\theta}}^{(m)}(\boldsymbol{\lambda})\|_2^2}\right) \tag{36}$$
>
> for $m = 1, ..., M$ from top left to bottom right. The Hessian is positive definite for any fixed $\epsilon > 0$. ✓

To calculate the gradient, we define the locally equivalent joint optimization problem, using the same notational shorthand $\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}$ and $\boldsymbol{X}_{V,I(\boldsymbol{\lambda})}$:

$$\begin{array}{c} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2_+} \frac{1}{2n}\left\|\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right\|_2^2 \\ \text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\left\|\boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})}\boldsymbol{\beta}\right\|_2^2 + \lambda_1\sum_{m=1}^M \|\boldsymbol{\beta}^{(m)}\|_2 + \lambda_2\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\epsilon\|\boldsymbol{\beta}\|_2^2 \end{array} \tag{37}$$

Since the training criterion is now smooth, we can take the gradient and set it to zero:

$$-\frac{1}{n}\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}^\top(\boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \lambda_1 \begin{bmatrix} \frac{\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})||_2} \\ ... \\ \frac{\hat{\boldsymbol{\beta}}^{(M)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(M)}(\boldsymbol{\lambda})||_2} \end{bmatrix} + \lambda_2 sgn(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \epsilon\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = 0 \tag{38}$$

From (9) and the chain rule, we get that the gradient of the validation loss is:

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X_V})) = -\frac{1}{n}\left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\frac{\partial}{\partial \boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right) \tag{39}$$

where

$$\frac{\partial}{\partial \boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \left(\frac{1}{n}\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} + \lambda_1 \boldsymbol{B}(\boldsymbol{\lambda}) + \epsilon \boldsymbol{I}_p\right)^{-1} \left[\begin{bmatrix} \frac{\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})||_2} \\ \dots \\ \frac{\hat{\boldsymbol{\beta}}^{(M)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(M)}(\boldsymbol{\lambda})||_2} \end{bmatrix} \quad sgn(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}))\right] \tag{40}$$

2.4.4 Generalized Lasso

The generalized lasso (Roth 2004) penalizes the ℓ_1 norm of the coefficients $\boldsymbol{\theta}$ weighted by some matrix $\boldsymbol{D}$. Depending on the choice of $\boldsymbol{D}$, the generalized lasso induces different structural constraints on the regression coefficients. Special cases include the fused lasso, trend filtering, and wavelet smoothing (Tibshirani et al. 2005), (Kim et al. 2009), (Donoho and Johnstone 1994).

To tune the regularization parameter λ, we formulate the generalized lasso as a joint optimization problem:

$$\begin{array}{c} \min_{\lambda \in \mathbb{R}_+} \frac{1}{2}\|\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\lambda)\|^2 \\ \text{s.t. } \hat{\boldsymbol{\theta}}(\lambda) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} \frac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\|^2 + \lambda\|\boldsymbol{D}\boldsymbol{\theta}\|_1 + \frac{1}{2}\epsilon\|\boldsymbol{\theta}\|_2^2 \end{array} \tag{41}$$

Let $I(\lambda)$ denote the indices of the zero elements of $\boldsymbol{D}\hat{\boldsymbol{\theta}}(\lambda)$:

$$I(\lambda) = \left\{ i | (\boldsymbol{D}\hat{\boldsymbol{\theta}}(\lambda))_i = 0 \text{ for } i = 1, ..., p \right\} \tag{42}$$

Let $\boldsymbol{I}_{I(\lambda)}$ be the submatrix of the $p \times p$ identity matrix consisting of columns with indices $I(\lambda)$. Since $\|\boldsymbol{D\theta}\|_1$ is differentiable in θ only along directions where the current zero elements of $\boldsymbol{D\theta}$ remain zero, the differentiable space S_λ is the null space of $\boldsymbol{I}_{I(\lambda)}^\top \boldsymbol{D}$:

$$S_\lambda = \mathcal{N}(\boldsymbol{I}_{I(\lambda)}^\top \boldsymbol{D}) \tag{43}$$

Let $\boldsymbol{U}_\lambda$ be an orthonormal basis for $\mathcal{N}(\boldsymbol{I}_{I(\lambda)}^\top \boldsymbol{D})$.

The first two conditions in Theorem 1 are satisfied by similar reasoning to that discussed in Section 2.4.2. For the third condition, we need to check that the Hessian matrix is invertible.

Condition 3: The Hessian matrix of $L_T(\cdot, \boldsymbol{\lambda})$ with respect to $\boldsymbol{U}_\lambda$ is

$$\boldsymbol{U}_\lambda^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{U}_\lambda + \epsilon \boldsymbol{U}_\lambda \tag{44}$$

This is positive definite for any fixed $\epsilon > 0$. ✓

Now we show the gradient calculations. Following Algorithm 3, we first define the locally equivalent joint optimization problem:

$$\begin{array}{c} \min_{\lambda \in \mathbb{R}_+} \frac{1}{2} \|\boldsymbol{y}_V - \boldsymbol{X}_V \boldsymbol{U}_\lambda \hat{\boldsymbol{\beta}}(\lambda)\|^2 \\ \text{s.t. } \hat{\boldsymbol{\beta}}(\lambda) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{U}_\lambda \boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{D}\boldsymbol{U}_\lambda \boldsymbol{\beta}\|_1 + \frac{1}{2}\epsilon \|\boldsymbol{U}_\lambda \boldsymbol{\beta}\|_2^2 \end{array} \tag{45}$$

Since the training criterion in (45) is differentiable with respect to $\boldsymbol{\beta}$, we have the gradient condition

$$-(\boldsymbol{X}_T \boldsymbol{U}_\lambda)^\top (\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{U}_\lambda \hat{\boldsymbol{\beta}}(\lambda)) + \lambda (\boldsymbol{D}\boldsymbol{U}_\lambda)^\top sgn(\boldsymbol{D}\boldsymbol{U}_\lambda \hat{\boldsymbol{\beta}}(\lambda)) + \epsilon \boldsymbol{U}_\lambda \hat{\boldsymbol{\beta}}(\lambda) = 0 \tag{46}$$

Implicit differentiation of (46) with respect to λ and solving for $\frac{\partial}{\partial \lambda} \hat{\boldsymbol{\beta}}(\lambda)$ gives us

$$\frac{\partial}{\partial \lambda} \hat{\boldsymbol{\beta}}(\lambda) = -(\boldsymbol{U}_\lambda^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{U}_\lambda + \epsilon \boldsymbol{U}_\lambda)^{-1} \boldsymbol{U}_\lambda^\top \boldsymbol{D}^\top sgn(\boldsymbol{D}\boldsymbol{U}_\lambda \hat{\boldsymbol{\beta}}(\lambda)) \tag{47}$$

Plugging in (47) to the chain rule gives us the gradient of the validation loss with respect to λ:

$$\nabla_\lambda L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\lambda)}(\boldsymbol{X_V})) = -\left(\boldsymbol{X}_V \boldsymbol{U}_\lambda \frac{\partial}{\partial \lambda} \hat{\boldsymbol{\beta}}(\lambda) \right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_V \boldsymbol{U}_\lambda \hat{\boldsymbol{\beta}}(\lambda) \right) \tag{48}$$

2.4.5 Additive Partially Linear Models

Now we consider an example from semi-parametric regression: an additive partially linear model (APLM) with Hodrick-Prescott (H-P) filtering for unevenly-spaced inputs and the lasso penalty. In APLMs, the response is modeled as the sum of nonlinear and linear functions. The combination of H-P filtering and lasso favors models with smooth non-parametric estimates and sparse linear effects.

In this example we have a measured a response y_i, a vector of "linearly modeled features" $\boldsymbol{x}_i$, and a single "continuously modeled feature" z_i for each of $i = 1, \ldots n$ observations. We believe that $\boldsymbol{y}$ can be modeled as an additive combination of these features:

$$\boldsymbol{y} = \boldsymbol{X}^\top \boldsymbol{\beta} + g(\boldsymbol{z}) + \epsilon \tag{49}$$

We want to estimate the coefficients $\boldsymbol{\beta}$ and values of the function g at our observations: $\boldsymbol{\theta} = (\theta_1, ..., \theta_n) \equiv (g(z_1), ..., g(z_n))$.

To formalize our optimization problem we give a bit of notation. Let $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ be the design matrix of the p linear predictors, $\boldsymbol{z} \in \mathbb{R}^n$ be the design vector for the nonlinear predictor, and $\boldsymbol{y} \in \mathbb{R}^n$ be the vector of responses. We assume that the observations are ordered such that $z_1 \le z_2 \le \cdots \le z_n$. Let $\boldsymbol{I}_T, \boldsymbol{I}_V$ be a matrices such that the training set $\boldsymbol{X}_T = \boldsymbol{I}_T\boldsymbol{X}$ and the validation set $\boldsymbol{X}_V = \boldsymbol{I}_V\boldsymbol{X}$.

Our joint optimization problem is defined as follows:

$$\begin{array}{c} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2_+} \frac{1}{2}\left\| \boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \boldsymbol{I}_V\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta},\boldsymbol{\theta}} \frac{1}{2}\left\| \boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\beta} - \boldsymbol{I}_T\boldsymbol{\theta} \right\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\lambda_2\|\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\epsilon\left(\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{\theta}\|_2^2\right) \end{array} \tag{50}$$

The second penalty in the training criterion $\|\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}\|_2^2$ is the H-P filter and penalizes second-order differences between the nonparametric estimates of $g(\boldsymbol{z})$. $\boldsymbol{D}(\boldsymbol{z})$ is defined as

$$\boldsymbol{D}(\boldsymbol{z}) = \boldsymbol{D}^{(1)} \cdot \text{diag}\left(\frac{1}{z_2 - z_1}, \frac{1}{z_3 - z_2}, ..., \frac{1}{z_n - z_{n-1}}, 0 \right) \cdot \boldsymbol{D}^{(1)} \tag{51}$$

where

$$\boldsymbol{D}^{(1)} = \begin{bmatrix} -1 & 1 & 0 & ... & 0 & 0 \\ 0 & -1 & 1 & ... & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & ... & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{52}$$

We again add an ϵ of ridge to ensure a differentiable hessian. Note that $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ in (50) must include estimates of $g(z_i)$ for z_i from the validation set. We accomplish this by including z_i values from both training and validation sets in constructing $\boldsymbol{D}(\boldsymbol{z})$.

In this example, the lasso is the only penalty which is not everywhere differentiable. Let the nonzero indices of $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ be denoted $I(\boldsymbol{\lambda}) = \{i | \hat{\beta}_i(\boldsymbol{\lambda}) \ne 0 \text{ for } i = 1, ..., p\}$. The differentiable space is then

$$S_{\boldsymbol{\lambda}} = \boldsymbol{C}(\boldsymbol{I}_{I(\boldsymbol{\lambda})}) \oplus \mathbb{R}^n \tag{53}$$

By the same reasoning as before, the first two conditions of Theorem 1 are satisfied. We now check for the third condition.

Condition 3: The Hessian matrix of $L_T(\cdot, \boldsymbol{\lambda})$ with respect to the basis

$$\begin{bmatrix} \boldsymbol{I}_{I(\boldsymbol{\lambda})} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_n \end{bmatrix} \tag{54}$$

is

$$H = \begin{bmatrix} \boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} + \epsilon \boldsymbol{I} & \boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{I}_T \\ \boldsymbol{I}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} & \boldsymbol{I}_T^\top \boldsymbol{I}_T + \lambda_2 \boldsymbol{D}(\boldsymbol{z})^\top \boldsymbol{D}(\boldsymbol{z}) + \epsilon \boldsymbol{I} \end{bmatrix} \tag{55}$$

The Hessian matrix is invertible for any $\lambda_2 > 0$ and any fixed $\epsilon > 0$.

We now calculate the gradient of the validation loss. Given $I(\boldsymbol{\lambda})$, the nonzero set of $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$, we define the locally equivalent joint optimization problem as

$$\begin{gathered} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \frac{1}{2} \left\| \boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) - \boldsymbol{I}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{s.t. } \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\eta}, \boldsymbol{\theta}} \frac{1}{2} \left\| \boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} \boldsymbol{\eta} - \boldsymbol{I}_T \boldsymbol{\theta} \right\|_2^2 + \lambda_1 \|\boldsymbol{\eta}\|_1 + \frac{1}{2}\lambda_2 \|\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\epsilon \left(\|\boldsymbol{\eta}\|_2^2 + \|\boldsymbol{\theta}\|_2^2 \right) \end{gathered} \tag{56}$$

As before we can now characterize our solution by setting the gradient of our now-locally-smooth optimization problem to 0. We then implicitly differentiate this gradient-based characterization and solve for $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda})$ and $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$. We get the following system of equations:

$$\begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \lambda_1} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_2} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \lambda_1} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_2} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) \end{bmatrix} = -H^{-1} \begin{bmatrix} sgn\left(\hat{\boldsymbol{\eta}}(\boldsymbol{\lambda})\right) & \boldsymbol{0} \\ \boldsymbol{0} & D^T(\boldsymbol{z}) D(\boldsymbol{z}) \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{bmatrix} \tag{57}$$

By the chain rule, the gradient of the validation loss is

$$\nabla_{\boldsymbol{\lambda}} L_V(\boldsymbol{\lambda}) = -\left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) + \boldsymbol{I}_V \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) - \boldsymbol{I}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right)$$

2.5 Additive model with heterogeneous smoothness

Finally, consider a nonparametric regression model for response y given p covariates $x_1, ..., x_p$

$$y = f(x_1, ..., x_p) + \epsilon \tag{58}$$

where ϵ are independent errors. When p is large, the multivariate function f is difficult to estimate due to the "curse of dimensionality." The additive model proposed by Buja et al.

(1989) addresses this problem by approximating the multivariate function with p univariate functions

$$y = \sum_{i=1}^{p} f_i(x_i) + \epsilon \tag{59}$$

This is also a special case of the PPR (projection pursuit regression) model proposed by Friedman and Stuetzle (1981), the ALS (alternating least squares) model of van der Burg and de Leeuw (1983) and the ACE (alternating conditional expectation) model of Breiman and Friedman (1985). (rewrite? remove? dunno?)

In this example, we fit smooth estimates $\boldsymbol{\theta}_i \in \mathbb{R}^n$ for $f_i(x_{i1}), f_i(x_{i2}), ..., f_i(x_{isn})$ using least squares with a ridge penalty for the second-order differences. Suppose there are a total of n observations, where the first $|T|$ observations are from the training set and the last $|V|$ observations are from the validation set. Accordingly, let $\boldsymbol{I}_T$ as the first $|T|$ rows and $\boldsymbol{I}_V$ as the last $|V|$ rows of an $n \times n$ identity matrix. The joint optimization problem is

$$\begin{array}{c} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^P} \frac{1}{2} \left\| y_V - \boldsymbol{I}_V \sum_{i=1}^p \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) \right\|_2^2 \\ \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) = \min_{\boldsymbol{\theta}^{(i)} \in \mathbb{R}^n} \frac{1}{2} \left\| \boldsymbol{y}_T - \boldsymbol{I}_T \sum_{i=1}^p \boldsymbol{\theta}^{(i)} \right\|_2^2 + \sum_{i=1}^p \lambda_i \left\| \boldsymbol{D}_{\boldsymbol{x}_i}^{(2)} \boldsymbol{\theta}^{(i)} \right\|_2^2 + \epsilon \|\boldsymbol{\theta}\|_2^2 \end{array} \tag{60}$$

The matrix $\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}$ gives the second-order differences between the nonparametric estimates of f_i. Let the n observations be ordered according to the ith covariate such that $x_{ik_1} \le x_{ik_2} \le ... \le x_{ik_n}$. $\boldsymbol{D}_{\boldsymbol{x}_i}^{(1)} \in \mathbb{R}^{n \times n}$ is the corresponding first-order difference matrix; so row $j = 1, ..., n-1$ of $\boldsymbol{D}_{\boldsymbol{x}_i}^{(1)}$ has a -1 in position k_j, 1 in position k_{j+1}, and 0 elsewhere and row n is all zeros. Then $\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}$ is

$$\boldsymbol{D}_{\boldsymbol{x}_i}^{(1)} \cdot \operatorname{diag}\left(\frac{1}{x_{ik_2} - x_{ik_1}}, \frac{1}{x_{ik_3} - x_{ik_2}}, ..., \frac{1}{x_{ik_n} - x_{ik_{n-1}}}, 0 \right) \cdot \boldsymbol{D}_{\boldsymbol{x}_i}^{(1)} \tag{61}$$

Again, the additional small, fixed ridge penalty in the training set criterion is to ensure strong convexity.

Note that the joint optimization problem gives the general case with tuning parameters $\lambda_1, ..., \lambda_p$. Usually, one combines them into a single regularization parameter λ since tuning $p \ge 2$ parameters is computationally intractable. In the simulation studies, we will demonstrate tuning all p parameters.

It is easy to check that all three conditions for Theorem 1 are satisfied. The problem is composed solely of quadratic functions, so the validation loss $L(\boldsymbol{y}_v, \boldsymbol{I}_V \sum_{i=1}^p \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}))$ differentiable everywhere with respect to $\boldsymbol{\lambda}$. Choosing $\epsilon > 0$ will satisfy the positive definite condition.

To calculate the gradient, we perform implicit differentiation with respect to λ_i on the KKT conditions for $\{\hat{\boldsymbol{\theta}}_i(\boldsymbol{\lambda})\}_{i=1,..,p}$. We find that

$$\begin{pmatrix} \frac{\partial}{\partial \lambda_i} \hat{\boldsymbol{\theta}}^{(1)}(\boldsymbol{\lambda}) \\ ... \\ \frac{\partial}{\partial \lambda_i} \hat{\boldsymbol{\theta}}^{(p)}(\boldsymbol{\lambda}) \end{pmatrix} = \boldsymbol{W}^{-1} \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D}_{\boldsymbol{x}_i}^{(2)\top} \boldsymbol{D}_{\boldsymbol{x}_i}^{(2)} \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) \\ \boldsymbol{0} \end{pmatrix} \tag{62}$$

where $\boldsymbol{W}$ is the matrix

$$\begin{pmatrix} \boldsymbol{I}_T^\top \boldsymbol{I}_T + \lambda_1 \boldsymbol{D}_{\boldsymbol{x}_1}^{(2)\top} \boldsymbol{D}_{\boldsymbol{x}_1}^{(2)} + \epsilon \boldsymbol{I} & \boldsymbol{I}_T^\top \boldsymbol{I}_T & ... & \boldsymbol{I}_T^\top \boldsymbol{I}_T \\ \boldsymbol{I}_T^\top \boldsymbol{I}_T & \boldsymbol{I}_T^\top \boldsymbol{I}_T + \lambda_2 \boldsymbol{D}_{\boldsymbol{x}_2}^{(2)\top} \boldsymbol{D}_{\boldsymbol{x}_2}^{(2)} + \epsilon \boldsymbol{I} & ... & \boldsymbol{I}_T^\top \boldsymbol{I}_T \\ ... & ... & ... & ... \\ \boldsymbol{I}_T^\top \boldsymbol{I}_T & \boldsymbol{I}_T^\top \boldsymbol{I}_T & ... & \boldsymbol{I}_T^\top \boldsymbol{I}_T + \lambda_p \boldsymbol{D}_{\boldsymbol{x}_p}^{(2)\top} \boldsymbol{D}_{\boldsymbol{x}_p}^{(2)} + \epsilon \boldsymbol{I} \end{pmatrix} \tag{63}$$

By the chain rule, the gradient of the validation loss is then

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y}_V, \boldsymbol{I}_V \sum_{i=1}^p \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda})) = -\left(\boldsymbol{I}_V \sum_{i=1}^p \frac{\partial}{\partial \lambda_i} \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) \right)^\top \left(\boldsymbol{y}_V - \boldsymbol{I}_V \sum_{i=1}^p \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) \right) \tag{64}$$

2.6 Gradient Descent Details

Here we discuss choice of step size and our convergence criterion.

There are many possible choices for our step size sequence $\{t^{(k)}\}$. Popular choices for convex problems are discussed in Boyd and Vandenberghe (2004). We chose a backtracking line search as discussed in Chapter 9. In our examples initial step size was between 0.5 and 1 and we backtrack with parameters $\alpha = 0.01$ and $\beta \in [0.01, 0.1]$. Details of backtracking line search are given in the Appendix. During gradient descent, it is possible that the step size will result in a negative regularization parameter; we reject any step that would set a regularization parameter to below a minimum threshold of $1e$-10.

Our convergence criterion is based on the change in our validation loss between iterates. More specifically, we stop our algorithm when

$$L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k+1)})}(\boldsymbol{X}_V)\right) - L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k)})}(\boldsymbol{X}_V)\right) \le \delta$$

for some prespecified tolerance δ. For the results in this manuscript we use $\delta = 1e\text{-}5$.

2.7 Accelerated Gradient Descent

We also use a modification of Algorithm 3 based on the work of Nesterov (1983). For smooth convex problems, these "accelerated" algorithms have faster worst-case convergence than gradient descent (while maintaining the same per-iteration complexity). In practice, these accelerated algorithms often vastly improve performance. In particular, we follow the recipe from O'Donoghue and Candes (2013) which performs adaptive restarts whenever the function value increases. As before, we choose step size using backtracking. We present the exact details in Algorithm 2 included in the Appendix.

3 Results: regressions with two regularization parameters

We ran two simulation studies for this paper. The purpose of this first set of simulations is to compare the performance and efficiency of grid-based and descent-based joint optimization across different regularization methods, namely the elastic net, sparse group lasso, and APLM.

The regularization parameters were tuned over a training/validation split. We implemented descent-based joint optimization using two different methods: gradient descent and accelerated gradient descent with adaptive restarts. For both grid search and descent-based joint optimization we fit the model over the training set using the splitting conic solver (SCS) in CVXPY (Diamond and Boyd 2016).

We describe the simulation settings for the three regression examples below, followed by a discussion of the results.

3.1 Elastic net

We generated thirty datasets, each consisting of 80 training and 20 validation observations with 250 predictors. The $\boldsymbol{x}_i$ were marginally distributed $N(\mathbf{0}, \boldsymbol{I})$ with $cor(x_{ij}, x_{ik}) = 0.5^{|j-k|}$.

The response vector $\boldsymbol{y}$ was generated from the model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \sigma\boldsymbol{\epsilon} \tag{65}$$

where

$$\boldsymbol{\beta} = (\underbrace{1, ..., 1}_{\text{size } 15}, \underbrace{0, ..., 0}_{\text{size } 235}) \tag{66}$$

and $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I})$. σ was chosen such that the signal to noise ratio is 2.

Both descent-based methods were initialized at (0.01, 0.01) and (10, 10). Grid search was performed over a 10 × 10 grid from 1e-5 to four times the largest eigenvalue of $\boldsymbol{X}_T^\top \boldsymbol{X}_T$.

3.2 Sparse group lasso

We generated thirty datasets, each consisting of 60 training and 15 validation observations with 1500 covariates. The predictors X were generated from a standard normal distribution. The response $\boldsymbol{y}$ was generated from the model

$$\boldsymbol{y} = \sum_{j=1}^{3} \boldsymbol{X}^{(j)}\boldsymbol{\beta}^{(j)} + \sigma\boldsymbol{\epsilon} \tag{67}$$

where $\boldsymbol{\beta}^{(j)} = (1, 2, 3, 4, 5, 0, ..., 0)$ for $j = 1, 2, 3$ and $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I})$. σ was chosen such that the signal to noise ratio was 2. For the sparse group lasso, we used $M = 150$ covariate groups with 10 covariates each.

Both descent-based methods were initialized at (0.01, 0.01), (1, 1), and (100, 100). Grid search was performed over a 10 × 10 grid from 1e-5 to $\max(\{||\boldsymbol{X}^{(j)T}\boldsymbol{y}||_2\}_{j=1,...,m})$.

3.3 Additive partially linear models

We generated thirty datasets, each consisting of 100 training and 25 validation observations with 20 linear predictors and one nonlinear predictor. Linear predictors were generated such that the first two groups of three features were highly correlated and the rest of the features were generated from a standard normal distribution:

$$\begin{aligned} x_{ij} &= Z_1 + \delta_{ij} \text{ for } j = 1, 2, 3 \\ x_{ij} &= Z_2 + \delta_{ij} \text{ for } j = 4, 5, 6 \\ x_{ij} &\sim N(0, 1) \text{ for } j = 7, ..., 20 \end{aligned} \tag{68}$$

Elastic Net		
	Validation Error	Runtime (sec)
Grid Search	0.34 (0.003)	10.74
Gradient Descent	0.34 (0.003)	4.43
Nesterov's Gradient Descent	0.34 (0.003)	2.28

Sparse Group Lasso		
	Validation Error	Runtime (sec)
Grid Search	1.36 (0.09)	161.29
Gradient Descent	1.36 (0.09)	71.34
Nesterov's Gradient Descent	1.36 (0.10)	67.10

APLM		
	Validation Error	Runtime (sec)
Grid Search	1.31 (0.05)	27.82
Gradient Descent	1.31 (0.05)	16.04
Nesterov's Gradient Descent	1.31 (0.05)	12.09

Table 2: Validation error comparisons for simulation studies in Section 3. Variance is provided in parentheses.

where $Z_1 \sim N(0,1)$, $Z_2 \sim N(0,1)$, and $\delta_{ij} \sim N(0, \frac{1}{16})$. Nonlinear predictors $\boldsymbol{z}$ were independently drawn the standard uniform distribution. The response $\boldsymbol{y}$ was generated from the model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \kappa g(\boldsymbol{z}) + \sigma\boldsymbol{\epsilon} \tag{69}$$

where $\boldsymbol{\beta} = (1,1,1,1,1,1,0,...,0)$ and $g(\boldsymbol{z}) = (2 - \boldsymbol{z})\sin(20\boldsymbol{z}^4)$. Constants κ and σ were chosen such that the linear to nonlinear ratio $\frac{||\boldsymbol{X}\boldsymbol{\beta}||_2}{||g(\boldsymbol{z})||_2}$ was 2 and the signal to noise ratio was 2.

Both descent-based methods were initialized at $\lambda_1 = \lambda_2 = 10^i$ for $i = -2, -1, 0, 1$. Grid search was performed over a 10×10 grid from $1e$-6 to 10.

3.4 Discussion of results

As shown in Table 2, the descent-based joint optimization and grid search have the same average performance in all three regression examples. In regards to computational time, descent-based joint optimization is only slightly faster than grid search. So for regressions with one or two regularization parameters, the two methods have the same performance and do not differ much in terms of runtime.

4 Results: regressions with more than two regularization parameters

In this second simulation study we tested descent-based joint optimization on regressions with more than two regularization parameters. The goal is to see if descent-based joint optimization is effective in a more complex model space and still run efficiently.

We experimented with generalizations of the simple regressions from the previous section. For the sparse group lasso, we tried an "un-pooled" version in which each covariate group has its own regularization parameter. For the additive partial linear model example, we added a ridge penalty as a third regularization term. We tuned these generalized regressions using descent-based joint optimization. Details regarding the joint optimization formulations and gradient derivations for these generalized regression models are in the Appendix.

We compared the performance of the fitted models from the generalized regressions to those from the original two-parameter regression on a separate test set. Our results show that the models from the generalized regressions achieved lower test error and tuning their regularization parameters using gradient descent was computationally tractable, even in cases with over a hundred regularization parameters.

4.1 Un-pooled sparse group lasso

We first generalize sparse group lasso by replacing the group lasso penalty parameter with individual penalty parameters for each covariate group. This "un-pooled" version of sparse group lasso is defined as follows:

$$\frac{1}{2n}\left\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\right\|_2^2 + \sum_{m=1}^{M} \lambda_1^{(m)} \|\boldsymbol{\theta}^{(m)}\|_2 + \lambda_2\|\boldsymbol{\theta}\|_1 \tag{70}$$

This version increases the number of penalty parameters from two to $M+1$, where M is the number of groups. The additional flexibility allows setting covariate and covariate group effects to zero by different thresholds. Hence un-pooled sparse group lasso may be better at modeling covariate groups with very different distributions.

We ran three experiments with different numbers of covariate groups M and total covariates p, as given in Table 3. The simulation settings were similar to the simulation settings and grid search procedure in Section 3.2. For gradient descent, the $M+1$ regularization parameters were initialized at $1e\text{-}4 \times \mathbf{1}^\top$, $1e\text{-}3 \times \mathbf{1}^\top$, and $1e\text{-}2 \times \mathbf{1}^\top$.

Model performance was assessed using three metrics: test error, β error (defined as $\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2^2$), and the percentage of nonzero coefficients correctly identified among all the true nonzero coefficients. The results show that un-pooled sparse group lasso tuned using gradient descent performed better by all metrics.

Gradient descent was significantly faster in all three settings. In fact, the runtimes for gradient descent did not grow as the number of regularization parameters increased.

4.2 Additive partially linear models with three regularization parameters

We generalized the APLM criterion in (50) by using the elastic net instead of the lasso penalty, as follows:

$$\frac{1}{2}\left\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\beta} - \boldsymbol{I}_T\boldsymbol{\theta}\right\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\lambda_2\|\boldsymbol{\beta}\|_2^2 + \frac{1}{2}\lambda_3\|\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}\|_2^2 \tag{71}$$

Since the elastic net tends to perform better when some of the predictors are correlated, we hypothesize that this generalized APLM is well-suited for cases where the linear predictors are correlated. We can test this since descent-based joint optimization makes tuning the regularization parameters computationally tractable.

We used the same simulation settings as those in Section 3.3. Gradient descent was initialized at $\boldsymbol{\lambda} = 10^i \times \mathbf{1}_3^\top$ for $i = -4, ..., 1$. We experimented with three nonlinear functions

n=60, p=300, g=3, M=30				
	β Error	% Correct Nonzero β	Test Error	Runtime (sec)
SGL	1.13	10.70	0.04	15.81
Un-pooled SGL	0.18	23.79	0.01	5.62

n=60, p=1500, g=3, M=50				
	β Error	% Correct Nonzero β	Test Error	Runtime (sec)
SGL	7.79	9.63	0.28	148.64
Un-pooled SGL	4.00	17.79	0.14	88.78

n=60, p=1500, g=3, M=150				
	β Error	% Correct Nonzero β	Test Error	Runtime (sec)
SGL	2.20	10.69	0.080	162.14
Un-pooled SGL	0.06	15.34	0.002	48.63

Table 3: Comparison of models from un-pooled sparse group lasso and sparse group lasso (SGL), tuned using gradient descent and grid search, respectively.

$g(z) = 4z^3 - z^2 + 2z$				
	β Error	θ Error	Test Error	Runtime (sec)
APLM 2	0.59	3.35	3.78	35.48
APLM 3	0.38	2.96	3.73	43.44

$g(z) = \sin(5z) + \sin(15(z-3))$				
	β Error	θ Error	Test Error	Runtime (sec)
APLM 2	0.51	3.76	3.90	37.04
APLM 3	0.34	3.73	3.79	45.95

$g(z) = (2-z)\sin(20z^4)$				
	β Error	θ Error	Test Error	Runtime (sec)
APLM 2	0.58	4.91	4.13	40.75
APLM 3	0.41	4.85	4.08	54.63

Table 4: Comparison of the performance of APLM with three penalties (APLM 3) and that with two penalties (APLM 2). The regularization parameters were tuned using gradient descent and grid search, respectively.

$g : \mathbb{R} \mapsto \mathbb{R}$ of varying levels of smoothness, as given in Table 4.

In addition to comparing models based on their test error, we measured the error of the fitted linear effects and the nonparametric estimates. These correspond to the β error ($||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}||_2^2$) and θ error ($||g(\boldsymbol{z}) - \boldsymbol{\theta}||_2^2$), respectively.

The results show that the generalized APLM criterion performed better by all three metrics. The linear model fits improved the most, which supports our hypothesis. Surprisingly, the estimation of the nonlinear components also improved slightly, even though the penalty term for the nonparametric estimates was not modified.

The runtime for tuning the three-parameter regularization problem was slightly longer than tuning the original two-parameter problem with grid search. Nonetheless, the runtime remained reasonable.

4.3 Additive models with heterogeneous smoothness

Finally, we consider the additive model from Section 2.5. This model is particularly useful when the functions $f_1, ..., f_p$ have varying levels of smoothness. Ideally, λ_i is large for f_i with nearly constant first-order derivatives and small for f_i with drastically changing first-order derivatives. While it is computationally intractable to tune the parameters via grid search for $p > 2$, gradient descent is computationally feasible and achieves good performance.

The simulation settings are as follows. We generated thirty datasets, each with 180 training, 60 validation, and 60 test observations with $p = 3$ covariates. Each set of covariates $\boldsymbol{x}_i$ was a random permutation of values $\delta_i - 15 + 0.1j$ for $j = 1, ..., 300$ where the random variable $\delta_i \sim \mathcal{U}(0, 0.1)$ jitters the start position. The functions are

$$\begin{aligned} f_1(x_1) &= 9\sin(2x_1) \\ f_2(x_2) &= x_2 \\ f_3(x_3) &= 6\cos(1.25x_3) + 6\sin(0.5x_3 + 0.5) \end{aligned} \tag{72}$$

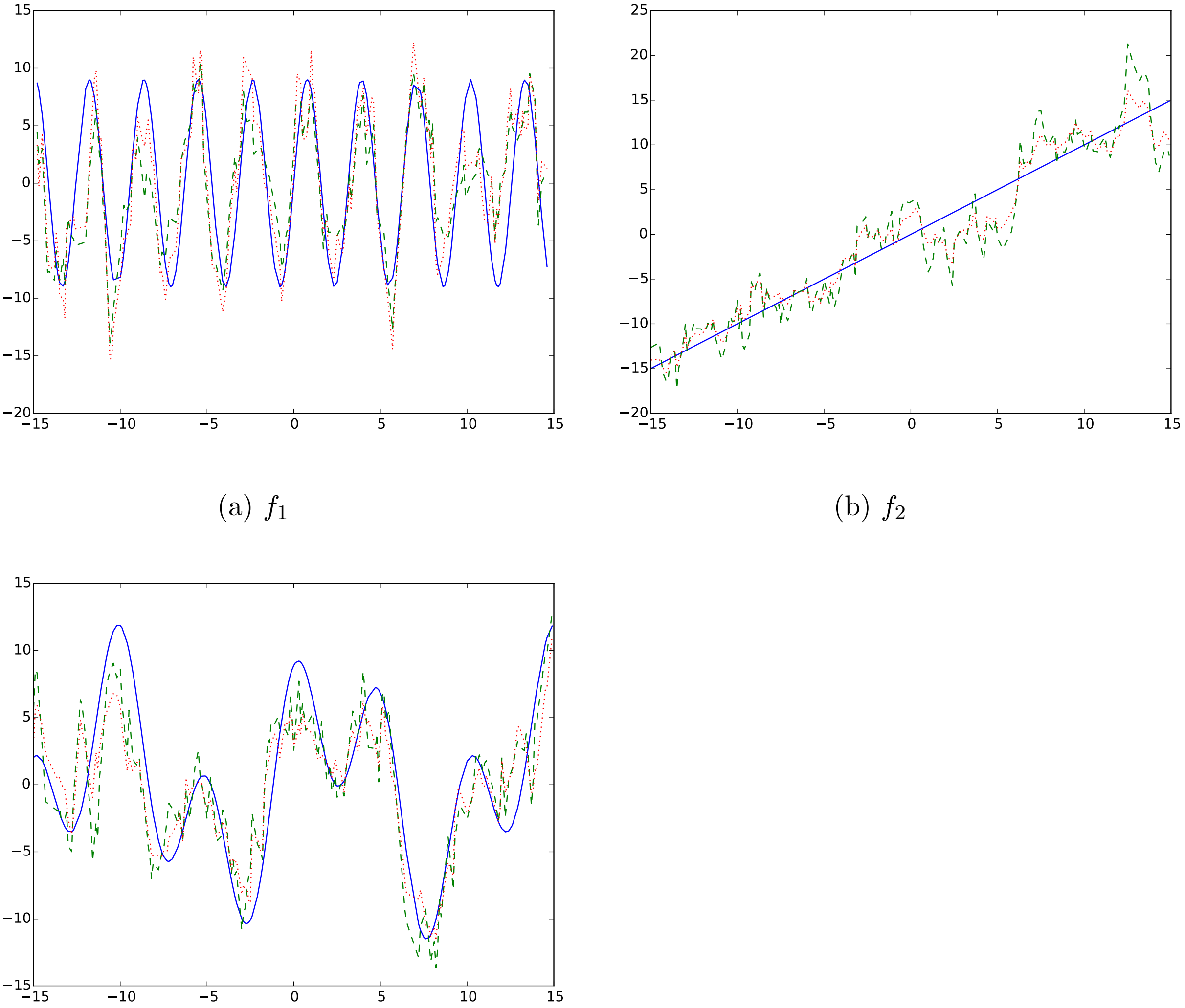
The response y was generated from the model

$$y = \sum_{i=1}^{3} f_i(x_i) + \sigma\epsilon \tag{73}$$

where $\epsilon \sim N(0, 1)$ and σ was chosen such that the signal to noise ratio was 2.

Since performing grid search for all 3 penalty parameters is computationally intractable, we did a one-dimensional search over 10 (log-spaced) values from $1e$-3 to 50. Gradient descent was initialized at $\lambda_1 = \lambda_2 = \lambda_3 = 1$. The inner optimization problem for the training set was solved using ECOS (Domahidi et al. 2013).

As seen in Table 5, the additive model with three penalty parameters has significantly lower validation error compared to one with a single penalty parameter, which is expected given the larger model space. Furthermore, the additive model with three penalty parameters achieved significantly lower test error. In Figure 2, we provide example estimates given by the model with one penalty parameter vs. three penalty parameters. As seen in the figures, the latter produces estimates with less variation for f_2 and more variation for f_1 and f_3, which is exactly what we would hope for. In fact, gradient descent consistently determined that f_2 was the smoothest function; for 27 of the thirty runs, $\hat{\lambda}_2$ was larger than $\hat{\lambda}_1$ and $\hat{\lambda}_3$.



(a) f_1

(b) f_2

(c) f_3

Figure 2: Example estimates $\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2, \hat{\boldsymbol{\theta}}_3$ for the test set: Left - Grid search, Right - Gradient descent. The solid lines are the true values of the test observations, the dotted lines are the estimates from the model with three penalty parameters, and the dashed lines are the estimates from the model with one penalty parameter.

	Validation Error	Test Error	Runtime (sec)
Gradient Descent	33.97 (1.31)	38.17 (1.61)	97.15
Grid Search	38.93 (1.57)	42.84 (1.88)	23.70

Table 5: Comparison of the performance of additive model with three penalty parameters tuned by gradient descent and that with one penalty parameter tuned by grid search. Standard errors are given in parentheses.

The runtime for tuning the additive model with three penalty parameters using gradient descent was slower than grid search, but still within a reasonable range. Gradient descent is certainly much faster than performing a three-dimensional grid search for this problem.

5 Application to Biological Data

Finally, we tested descent-based joint optimization in a real data example. More specifically, we considered the problem of finding predictive genes from gene pathways for Crohn's Disease and Ulcerative Colitis. Simon et al. (2013) addressed this problem using the sparse group lasso; we now compare this against applying the un-pooled sparse group lasso, where the regularization parameters were tuned using gradient descent.

Our dataset is from a colitis study of 127 total patients, 85 with colitis (59 crohn's patients + 26 ulcerative colitis patients) and 42 healthy controls (Burczynski et al. 2006). Expression data was measured for 22,283 genes on affymetrix U133A microarrays. We grouped the genes according to the 326 C1 positional gene sets from MSigDb v5.0 (Subramanian et al. 2005) and discarded the 2358 genes not found in the gene set.

We randomly shuffled the data and used the first 50 observations for the training set and the remaining 77 for the test set. Five-fold cross validation was used to fit models. To tune the penalty parameters in un-pooled sparse group lasso, we initialized gradient descent at $0.5 \times \mathbf{1}^\top$. For sparse group lasso, we tuned the penalty parameters over a 5×5 grid $1e$-4 to 5.

Table 6 presents the average results from repeating this process ten times. Un-pooled sparse group lasso achieved a slightly higher classification rate than sparse group lasso.

	% Correct	Num. Genesets	Num. Genes	Runtime (sec)
SGL	82.47 (0.7)	38.4 (671.2)	207.0 (22206.2)	2722.4
Un-pooled SGL	84.29 (0.3)	8.9 (1.9)	83.9 (664.5)	2298.5

Table 6: Comparison of predictive genes and genesets of Ulcerative Colitis found by un-pooled sparse group lasso and sparse group lasso (SGL). The variance is given in parenthesis.

Interestingly, un-pooled sparse group lasso found solutions that were significantly more sparse than sparse group lasso; on average, un-pooled sparse group lasso identified 9 genesets whereas sparse group lasso identified 38. These results suggest that un-pooling the penalty parameters in sparse group lasso could potentially improve interpretability.

In regards to runtime, we find that descent-based joint optimization for un-pooled sparse group lasso was computationally tractable, even though it required tuning 327 regularization parameters. In fact, it was slightly faster than grid-based joint optimization for sparse group lasso.

6 Discussion

In this paper, we proposed finding the optimal regularization parameters by treating it as an optimization problem over the regularization parameter space. We have proven that a descent-based approach can be used for regression problems in which the penalties are smooth almost everywhere and present a general algorithm for performing a modified gradient descent.

Empirically, we find that models fit by descent-based joint optimization have similar accuracy to those from grid search. Furthermore, the scalability of this approach allows us to test new regression problems with multiple penalties. In particular, we found that an un-pooled variant of sparse group lasso showed promising results. More research should be done to explore this new regularization method.

Future work could include finding other classes of regularization methods that are suitable for descent-based joint optimization and implementing descent-based joint optimization with more sophisticated optimization methods.

References

Boyd, S. & Vandenberghe, L. (2004), *Convex optimization*, Cambridge university press.

Buja, A., Hastie, T. & Tibshirani, R. (1989), 'Linear smoothers and additive models', *The Annals of Statistics* pp. 453–510.

Burczynski, M. E., Peterson, R. L., Twine, N. C., Zuberek, K. A., Brodeur, B. J., Casciotti, L., Maganti, V., Reddy, P. S., Strahs, A., Immermann, F. et al. (2006), 'Molecular classification of crohn's disease and ulcerative colitis patients using transcriptional profiles in peripheral blood mononuclear cells', *The journal of molecular diagnostics* **8**(1), 51–61.

Diamond, S. & Boyd, S. (2016), 'Cvxpy: A python-embedded modeling language for convex optimization', *Journal of Machine Learning Research* . To appear.
URL: *http://stanford.edu/ boyd/papers/pdf/cvxpy_paper.pdf*

Domahidi, A., Chu, E. & Boyd, S. (2013), ECOS: An SOCP solver for embedded systems, *in* 'European Control Conference (ECC)', pp. 3071–3076.

Donoho, D. L. & Johnstone, J. M. (1994), 'Ideal spatial adaptation by wavelet shrinkage', *Biometrika* **81**(3), 425–455.

Golub, G. H., Heath, M. & Wahba, G. (1979), 'Generalized cross-validation as a method for choosing a good ridge parameter', *Technometrics* **21**(2), 215–223.

Kim, S.-J., Koh, K., Boyd, S. & Gorinevsky, D. (2009), '\ell_1 trend filtering', *SIAM review* **51**(2), 339–360.

Lorbert, A. & Ramadge, P. J. (2010), Descent methods for tuning parameter refinement, *in* 'International Conference on Artificial Intelligence and Statistics', pp. 469–476.

Mammen, E., van de Geer, S. et al. (1997), 'Locally adaptive regression splines', *The Annals of Statistics* **25**(1), 387–413.

Nesterov, Y. (1983), A method of solving a convex programming problem with convergence rate o (1/k2), Vol. 27, Soviet Mathematics Doklady, pp. 372–376.

O'Donoghue, B. & Candes, E. (2013), 'Adaptive restart for accelerated gradient schemes', *Foundations of computational mathematics* **15**(3), 715–732.

Roth, V. (2004), 'The generalized lasso', *Neural Networks, IEEE Transactions on* **15**(1), 16–28.

Simon, N., Friedman, J., Hastie, T. & Tibshirani, R. (2013), 'A sparse-group lasso', *Journal of Computational and Graphical Statistics* **22**(2), 231–245.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. et al. (2005), 'Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles', *Proceedings of the National Academy of Sciences of the United States of America* **102**(43), 15545–15550.

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

Tsybakov, A. (2008), *Introduction to Nonparametric Estimation*, Springer Series in Statistics, Springer.
URL: *https://books.google.com/books?id=mwB8rUBsbqoC*

Wahba, G. (1981), 'Spline interpolation and smoothing on the sphere', *SIAM Journal on Scientific and Statistical Computing* **2**(1), 5–16.

Wood, S. N. (2000), 'Modelling and smoothing parameter estimation with multiple quadratic penalties', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **62**(2), 413–428.

Zou, H. & Hastie, T. (2003), 'Regression shrinkage and selection via the elastic net, with applications to microarrays', *Journal of the Royal Statistical Society: Series B. v67* pp. 301–320.