Descent-based joint optimization

Jean Feng and Noah Simon

Abstract

Tuning regularization parameters in regression problems allows one to control model complexity and induce desired structure. The current method of searching over a k-dimensional grid of parameter values is computationally intractable for $k > 2$. We propose tuning the penalty parameters by treating it as a continuous optimization problem and updating the parameter values using a descent-based approach. Compared to performing cross validation over a grid, our method is significantly more efficient while achieving the same performance. This descent-based approach enables us to test regularizations with many penalty parameters, through which we discover new regularization methods with superior accuracy. Our experiments are performed on simulated and real data.

1 Introduction

Consider the usual regression framework with p features, $x_{i1}, \ldots, x_{ip}$ and a response y_i measured on each of $i = 1, \ldots, n$ observations. Let X denote the $n \times p$ design matrix and y the response vector. Our goal here is to characterize the conditional relationship between y and 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 y and $f(X)$ (generally quantified with some pre-specified loss, L). Often $\mathcal{F}$ will endow f with some simple form (eg. 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:

$$\hat{f}(\lambda_1, ..., \lambda_J) = \underset{f \in \mathcal{F}}{\arg\min}\, L(\boldsymbol{y}, f(\boldsymbol{X})) + \sum_{i=1}^{J} \lambda_i P_i(f) \tag{1}$$

where $\{P_i\}_{i=1,\ldots,J}$ are the penalty functions, and $\{\lambda_i\}_{i=1,\ldots,J}$ are the regularization parameters.

Regularization parameters control the degree of various facets of model complexity (e.g. 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 $(X_T, \boldsymbol{y}_T)$ and measures the model's error on a validation set $(X_V, \boldsymbol{y}_V)$. The goal then is to choose penalty parameters, $\lambda_1, \ldots, \lambda_J$, that minimize the validation error, as formulated in the following optimization problem:

$$\begin{gathered}\min_{\boldsymbol{\lambda} \in \Lambda} L(\boldsymbol{y}_V, \hat{f}(\boldsymbol{X}_V|\boldsymbol{\lambda})) \\ \text{where } \hat{f}(\cdot|\boldsymbol{\lambda}) = \arg\min_{f \in \mathcal{F}} L(\boldsymbol{y}_T, f(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(f)\end{gathered} \tag{2}$$

Here Λ is some set that $\boldsymbol{\lambda}$ are known to be in (possibly just $\mathbb{R}^{n+}$).

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. Many variants of grid search have been proposed to increase efficiency, but their runtimes are all exponential in the number of parameters.

In this paper, we propose leveraging the tools of optimization to solve (2) over the penalty parameter space. We give a gradient descent algorithm for the penalty parameters (to minimize validation error). 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. In particular this paper focuses on three examples that demonstrate the wide applicability of our method: elastic net, sparse group lasso, and additive partially linear models.

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 4, 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, $\mathcal{F}$, which, for a fixed n, are finite dimensional; ie. can be rewritten as $\mathcal{F} = \{f_\theta | \theta \in \Theta\}$ for some finite dimensional 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 (eg. 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:

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

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

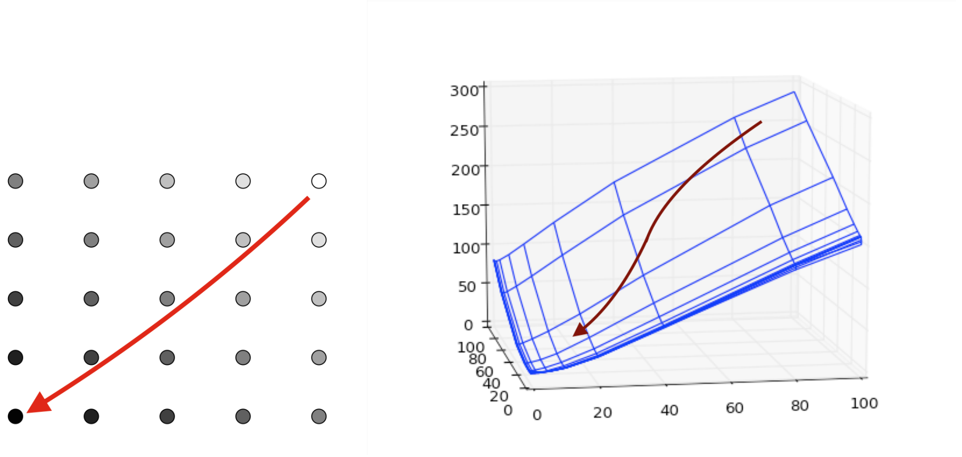

Figure 1. Left: darker points mean lower validation loss. Descent-based optimization descends in the most direct path towards the point producing the lowest validation loss. Right: The 3D version. We can tune regularization parameters using a grid search... or just descend opposite of the gradient.

joint optimization problem (2) over this finite-dimensional class as:

$$\begin{aligned} &\arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^J_+} L(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ \text{where } \hat{\theta}(\boldsymbol{\lambda}) = &\arg\min_{\theta \in \Theta} L(\boldsymbol{y}_T, f_\theta(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\theta) \end{aligned} \tag{4}$$

For the remained of the manuscript we will assume that (3) is strictly convex in θ. This ensures that there is a unique solution to (3) which perturbs continuously in $\boldsymbol{\lambda}$.

(4) is the explicit, though often unstated, criterion that training/validation methods attempt to minimize to choose 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 G in the penalty parameter space — the computational runtime of grid-based methods grows exponentially with the number of 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. Using an exhaustive search ignores information available from the smoothness of the surface.

We propose leveraging the tools of smooth optimization to solve (4). 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 applied in situations where the penalty is only differentiable when restricted to directions involving a differentiable set.

Figure 1 illustrates the differences between the two approaches. Grid-based method fits a model at every grid point, even though many of these grid points are not close to the global or local minima. We can save significant computational time if we avoid those points unlikely to yield good models. By incorporating information about the shape of the local neighborhood, descent-based methods choose an intelligent descent direction and explore the space more efficiently.

Of course, the joint optimization problem is non-convex and therefore our method provides no guarantees, though, in practice, we have seen strong empirical performance. The major benefit of using our method is that it opens up the possibility of using regularization methods that combine many penalty terms.

To ease exposition, we will assume throughout the remainder of the manuscript that $L\Big(\boldsymbol{y}_V, f_\theta(\boldsymbol{X}_V)\Big)$

is differentiable in θ. This assumption is met if both 1) $f_\theta(\boldsymbol{X}_V)$ is continuous as a function of θ; 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

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

First we consider the simple case where $L_T\left(\theta, \boldsymbol{\lambda}\right)$ is smooth as a function of $(\theta, \boldsymbol{\lambda})$. As shown later, the validation loss is differentiable as a function of $\boldsymbol{\lambda}$. So, we can directly apply gradient descent to find a local minimum of our criterion (4), as described in Algorithm 1.

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 $t^{(k)}$
        $$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - t^{(k)} \nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)\Big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} \tag{6}$$
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 $t^{(k)} = t/k$.

Calculating the Gradient: To find the gradient, we first apply the chain rule:

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

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

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

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

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

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

We can plug (9) into (7) to get $\nabla_{\boldsymbol{\lambda}} L\Big(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\Big)$. Note that because $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda})$ is defined in terms of $\hat{\theta}\left(\boldsymbol{\lambda}\right)$, each gradient step requires solving the penalized regression problem (3) on the training data. Algorithm 2 is the updated version of Algorithm 1 with the specific gradient calculations.

Algorithm 2 Updated Algorithm 1

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

2.3 Nonsmooth Training Criterion

When the penalized training criterion in the joint optimization problem is not smooth, gradient descent cannot be applied directly. Nonetheless, the solution $\hat{\theta}\left(\boldsymbol{\lambda}\right)$ is often still smooth at almost every $\boldsymbol{\lambda}$ (eg. Lasso, Group Lasso, Trend Filtering). Thus we can calculate $\nabla_{\boldsymbol{\lambda}} L\Big(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\Big)$ using (7) and apply gradient descent in practice. In this section, we provide a general approach for solving such problems that addresses the two primary challenges: characterizing problems for which $\hat{\theta}\left(\boldsymbol{\lambda}\right)$ is almost everywhere smooth and calculating $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda})$.

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

Definition 1. *The differentiable space of a real-valued function L and a point η in its domain is defined as the set of vectors u such that the directional derivative of L at η along u exists.*

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

Definition 2. *S is defined as a local optimality space for a convex function $L(\cdot, \lambda_0)$ if there exists a neighborhood W containing λ_0 such that for every $\lambda \in W$,*

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

Definition 3. *Let $B = \{b_1, ..., b_p\}$ be an orthonormal set of vectors in $\mathbb{R}^n$. 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 vectors in B exist. The Gradient vector and Hessian matrix of f with respect to B are defined respectively as*

$$_B\nabla f \in \mathbb{R}^p = \begin{pmatrix} \frac{\partial f}{\partial b_1} \\ \frac{\partial f}{\partial b_2} \\ \vdots \\ \frac{\partial f}{\partial b_p} \end{pmatrix}; \quad _B\nabla^2 f \in \mathbb{R}^{p \times p} = \begin{pmatrix} \frac{\partial^2 f}{\partial b_1^2} & \frac{\partial^2 f}{\partial b_1 \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_1 \partial b_p} \\ \frac{\partial^2 f}{\partial b_2 \partial b_1} & \frac{\partial^2 f}{\partial b_2^2} & \cdots & \frac{\partial^2 f}{\partial b_2 \partial b_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial b_p \partial b_1} & \frac{\partial^2 f}{\partial b_p \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_p^2} \end{pmatrix} \tag{15}$$

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

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

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

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

Note that condition 3 actually implies that the Hessian of $L_T(\cdot,\boldsymbol{\lambda})$ with respect to any orthonormal basis of $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))$ 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 $L_T(\boldsymbol{\theta},\boldsymbol{\lambda})$ 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 $L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X_V}))$ is continuously differentiable with respect to $\boldsymbol{\lambda}$ for almost every $\boldsymbol{\lambda}$. Furthermore, the gradient of $L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V}))$, where it is defined, is

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

where

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

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

$$\begin{gathered}\min\nolimits_{\boldsymbol{\lambda}\in\mathbb{R}^J_+} L(y_V, f_{U^{(k)}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(X_V)) \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min\nolimits_{\boldsymbol{\beta}} L(y_T, f_{U^{(k)}\boldsymbol{\beta}}(X_T)) + \sum_{i=1}^{J} \lambda_i P_i(U\boldsymbol{\beta})\end{gathered} \tag{18}$$

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

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

Note that because the differentiable space is a local optimality space and is thus locally constant, we can treat U as a constant in the gradient derivations. In Algorithm 3, we give the descent procedure in more detail.

To perform joint optimization for full K-fold cross validation, the objective simply become the average validation cost across all K folds. 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 $(\boldsymbol{y}_{-k},\boldsymbol{X}_{-k})$. The joint optimization problem then becomes

$$\begin{gathered}\arg\min\nolimits_{\boldsymbol{\lambda}\in\mathbb{R}^J_+} \tfrac{1}{K}\sum\nolimits_{k=1}^{K} L(\boldsymbol{y}_k, f_{\hat{\theta}^{(k)}(\boldsymbol{\lambda})}(\boldsymbol{X}_k)) \\ \text{where } \hat{\theta}^{(k)}(\boldsymbol{\lambda}) = \arg\min\nolimits_{\theta\in\Theta} L(\boldsymbol{y}_{-k}, f_\theta(\boldsymbol{X}_{-k})) + \sum_{i=1}^{J} \lambda_i P_i(\theta) \text{ for } k = 1, ..., K\end{gathered} \tag{23}$$

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 $\hat{\theta}(\boldsymbol{\lambda}^{(k)}) = \arg\min_{\theta \in \mathbb{R}^p} L_T(\theta, \boldsymbol{\lambda}^{(k)})$.
    Construct matrix $U^{(k)}$, an orthonormal basis of $\Omega^{L_T(\cdot, \boldsymbol{\lambda})}\left(\hat{\theta}\left(\boldsymbol{\lambda}^{(k)}\right)\right)$.
    Define the locally equivalent joint optimization problem

        $$\begin{gathered} \min_{\boldsymbol{\lambda}} L(y_V, f_{U^{(k)}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(X_V)) \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} L(y_T, f_{U^{(k)}\boldsymbol{\beta}}(X_T)) + \sum_{i=1}^{J} \lambda_i P_i(U^{(k)}\boldsymbol{\beta}) \end{gathered} \tag{20}$$

    Calculate $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\beta}(\boldsymbol{\lambda})|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}}$ where

        $$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\left[ {}_{U^{(k)}}\nabla^2 \left( L(\boldsymbol{y}_T, f_{U^{(k)}\boldsymbol{\beta}}(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(U^{(k)}\boldsymbol{\beta}) \right) \Bigg|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right]^{-1} \left[ {}_{U^{(k)}}\nabla P(U^{(k)}\boldsymbol{\beta}) \Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right] \tag{21}$$

    with ${}_{U^{(k)}}\nabla^2$ and ${}_{U^{(k)}}\nabla$ 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

        $$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_{\boldsymbol{V}}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_{\boldsymbol{V}})\right) = \left[ U^{(k)} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right]^{\top} \left[ {}_{U^{(k)}}\nabla L\left(\boldsymbol{y}_{\boldsymbol{V}}, f_{U^{(k)}\boldsymbol{\beta}}(\boldsymbol{X}_{\boldsymbol{V}})\right)|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right] \tag{22}$$

    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
```

	Differentiable Space
Ridge Regression	$\mathbb{R}^p$
Elastic Net	$span(\{e_i \| \hat{\theta}_i(\boldsymbol{\lambda}) \neq 0\})$
Sparse Group Lasso	$span(\{e_i \| \hat{\theta}_i(\boldsymbol{\lambda}) \neq 0\})$
Generalized Lasso	$\mathcal{N}(I_{I(\lambda)}D)$ where $I(\lambda) = \{i \| (D\hat{\boldsymbol{\theta}}(\lambda))^\top e_i = 0\}$
Additive Partially Linear Model	$span(\{e_i \| \hat{\beta}_i(\boldsymbol{\lambda}) \neq 0\}) \oplus \mathbb{R}^n$

Table 1. The differentiable space of each example regression problem

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 $\hat{\theta}(\boldsymbol{\lambda})$ is smooth almost everywhere. Finally, we discuss doing descent-based joint optimization for an additive partially linear model, an example of a semi-parametric regression.

In all of these problems, for almost every $\boldsymbol{\lambda}$, the differentiable space of $L_T(\cdot, \boldsymbol{\lambda})$ exists, and is also a local optimality space. For reference, the differentiable space for each regression example is specified in Table 1. For ease of notation, we will let $S_{\boldsymbol{\lambda}}$ denote the differentiable space of $L_T(\cdot, \boldsymbol{\lambda})$ at $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$.

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.

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

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:

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

Let the nonzero indices of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ be denoted $I(\boldsymbol{\lambda}) = \{i|\hat{\boldsymbol{\theta}}_i(\boldsymbol{\lambda}) \neq 0\}$ and let $I_{I(\boldsymbol{\lambda})}$ 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 $I_{I(\boldsymbol{\lambda})}$. That is, the differentiable space at $\boldsymbol{\lambda}$ is

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

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 $I_{I(\boldsymbol{\lambda})}$ is $I_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T I_{I(\boldsymbol{\lambda})} + \lambda_2 \boldsymbol{I}$. This is positive definite if $\lambda_2 > 0$. ✓

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

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

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

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

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 (eg. 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.

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

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 $I(\boldsymbol{\lambda}) = \{i | \hat{\boldsymbol{\theta}}_i(\boldsymbol{\lambda}) \neq 0\}$ are the nonzero indices of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$.

This problem satisfies all three conditions in Theorem 1. Since the logic 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 $I_{I(\boldsymbol{\lambda})}$ is

$$\frac{1}{n} I_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T I_{I(\boldsymbol{\lambda})} + \lambda_1 \boldsymbol{B}(\boldsymbol{\lambda}) + \epsilon \boldsymbol{I}_p \tag{35}$$

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 $\boldsymbol{X}_{V,I(\boldsymbol{\lambda})}$:

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

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:

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

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

2.4.4 Generalized Lasso

The generalized lasso [Roth, 2004] penalizes the ℓ_1 norm of the coefficients $\boldsymbol{\theta}$ weighted by some matrix D. Depending on the choice of 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:

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

Let $I(\lambda)$ denote the indices of the zero elements of $D\hat{\boldsymbol{\theta}}(\lambda)$, ie $I(\lambda) = \left\{ i | (D\hat{\boldsymbol{\theta}}(\lambda))_i = 0 \right\}$. Let $I_{I(\lambda)}$ be the submatrix of the $p \times p$ identity matrix consisting of columns with indices $I(\lambda)$. Since $\|D\boldsymbol{\theta}\|_1$ is differentiable in θ only along directions where the current zero elements of $D\boldsymbol{\theta}$ remain zero, the differentiable space S_λ is the null space of $I_{I(\lambda)}^\top D$:

$$S_\lambda = \mathcal{N}(I_{I(\lambda)}^\top D) \tag{42}$$

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

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 U_λ is

$$U_\lambda^\top X_T^\top X_T U_\lambda + \epsilon U_\lambda \tag{43}$$

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:

$$\begin{array}{c} \min_{\lambda \in \mathbb{R}_+} \frac{1}{2} \|\boldsymbol{y}_V - \boldsymbol{X}_V U_\lambda \hat{\boldsymbol{\beta}}(\lambda)\|^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\lambda) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_T U_\lambda \boldsymbol{\beta}\|^2 + \lambda \|D U_\lambda \boldsymbol{\beta}\|_1 + \frac{1}{2}\epsilon \|U_\lambda \boldsymbol{\beta}\|_2^2 \end{array} \tag{44}$$

Since the training criterion in (44) is differentiable with respect to $\boldsymbol{\beta}$, we have the gradient condition

$$-(\boldsymbol{X}_T U_\lambda)^\top (\boldsymbol{y}_T - \boldsymbol{X}_T U_\lambda \hat{\boldsymbol{\beta}}(\lambda)) + \lambda (D U_\lambda)^\top sgn(D U_\lambda \hat{\boldsymbol{\beta}}(\lambda)) + \epsilon U_\lambda \hat{\boldsymbol{\beta}}(\lambda) = 0 \tag{45}$$

Implicit differentiation of (45) with respect to λ and solving for $\frac{\partial}{\partial \lambda} \hat{\boldsymbol{\beta}}(\lambda)$ gives us

$$\frac{\partial}{\partial \lambda} \hat{\boldsymbol{\beta}}(\lambda) = -(U_\lambda^\top X_T^\top X_T U_\lambda + \epsilon U_\lambda)^{-1} U_\lambda^\top D^\top sgn(D U_\lambda \hat{\boldsymbol{\beta}}(\lambda)) \tag{46}$$

Plugging in (46) to the chain rule gives us the gradient of the validation loss with respect to λ:

$$\nabla_\lambda L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\lambda)}(\boldsymbol{X_V})) = -\left(\boldsymbol{X}_V U_\lambda \frac{\partial}{\partial \lambda} \hat{\boldsymbol{\beta}}(\lambda) \right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_V U_\lambda \hat{\boldsymbol{\beta}}(\lambda) \right) \tag{47}$$

2.4.5 Additive Partially Linear Models

Finally, we present 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 model fits with smooth nonparametric estimates and sparse linear effects.

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

$$y_i = \boldsymbol{x_i}^\top \boldsymbol{\beta} + g(z_i) + \epsilon_i \tag{48}$$

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 $z_1 \leq z_2 \leq \cdots \leq z_n$. Let $\boldsymbol{I}_T$ be a $n \times n$ diagonal matrix with elements 1 or 0. $\boldsymbol{I}_T$ partitions the data into the training set $\boldsymbol{X}_T = \boldsymbol{I}_T \boldsymbol{X}$ and the validation set $\boldsymbol{X}_V = (\boldsymbol{I} - \boldsymbol{I}_T)\boldsymbol{X}$.

Our joint optimization problem is defined as follows:

$$\begin{array}{c} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \frac{1}{2} \left\| \boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - (\boldsymbol{I} - \boldsymbol{I}_T)\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}, \boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\beta} - \boldsymbol{I}_T \boldsymbol{\theta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \frac{1}{2}\lambda_2 \|\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\epsilon \left(\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{\theta}\|_2^2 \right) \end{array} \tag{49}$$

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

$$\boldsymbol{D}(\boldsymbol{z}) = \boldsymbol{D}^{(1)} \cdot \mathrm{diag}\left(\frac{1}{z_2 - z_1}, \frac{1}{z_3 - z_2}, ..., \frac{1}{z_n - z_{n-1}}, 0\right) \cdot \boldsymbol{D}^{(1)} \tag{50}$$

where

$$\boldsymbol{D}^{(1)} = \begin{bmatrix} -1 & 1 & 0 & ... & 0 & 0 \\ 0 & -1 & 1 & ... & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & ... & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{51}$$

We again add an ϵ of ridge to ensure a differentiable hessian. Note that $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ in (49) 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 part which is not everywhere differentiable. Let the nonzero indices of $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ be denoted $I(\boldsymbol{\lambda}) = \{i|\hat{\boldsymbol{\beta}}_i(\boldsymbol{\lambda}) \neq 0\}$. The differentiable space is then

$$S_{\boldsymbol{\lambda}} = C(\boldsymbol{I}_{I(\boldsymbol{\lambda})}) \oplus \mathbb{R}^n \tag{52}$$

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

$$\begin{bmatrix} \boldsymbol{I}_{I(\boldsymbol{\lambda})} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_n \end{bmatrix} \tag{53}$$

is

$$H = \begin{bmatrix} \boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} + \epsilon \boldsymbol{I} & \boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{I}_T \\ \boldsymbol{I}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} & \boldsymbol{I}_T^\top \boldsymbol{I}_T + \lambda_2 \boldsymbol{D}(\boldsymbol{z})^\top \boldsymbol{D}(\boldsymbol{z}) + \epsilon \boldsymbol{I} \end{bmatrix} \tag{54}$$

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

$$\begin{gathered} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \tfrac{1}{2} \left\| \boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) - (\boldsymbol{I} - \boldsymbol{I}_T)\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{where } \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\eta},\boldsymbol{\theta}} \tfrac{1}{2} \left\| \boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} \boldsymbol{\eta} - \boldsymbol{I}_T \boldsymbol{\theta} \right\|_2^2 + \lambda_1 \|\boldsymbol{\eta}\|_1 + \tfrac{1}{2}\lambda_2 \|\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}\|_2^2 + \tfrac{1}{2}\epsilon \left(\|\boldsymbol{\eta}\|_2^2 + \|\boldsymbol{\theta}\|_2^2 \right) \end{gathered} \tag{55}$$

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:

$$\begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \lambda_1} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_2} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \lambda_1} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_2} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) \end{bmatrix} = -H^{-1} \begin{bmatrix} sgn\left(\hat{\boldsymbol{\eta}}(\boldsymbol{\lambda})\right) & \boldsymbol{0} \\ \boldsymbol{0} & D^T(\boldsymbol{z}) D(\boldsymbol{z}) \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{bmatrix} \tag{56}$$

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

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

2.5 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 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

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

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

2.6 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 5 included in the Appendix.

3 Results: validation error minimization

We ran two simulation studies for this paper. The purpose of the 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. The inner optimization problem in descent-based and grid-based joint optimization were solved 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 x_i were marginally distributed $N(0, 1)$ with $cor(x_i, x_j) = 0.5^{|i-j|}$. The response vector y was generated from the model

$$y = X\beta + \sigma\epsilon \tag{57}$$

where

$$\beta = (\underbrace{1, ..., 1}_{\text{size } 15}, \underbrace{0, ..., 0}_{\text{size } 235}) \tag{58}$$

and $\epsilon \sim N(0, 1)$. 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-based joint optimization searched over a 10×10 grid from $1e$-5 to four times the largest eigenvalue of $X_T^\top X_T$.

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 (variance in parentheses)

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 y was generated from the model

$$y = \sum_{j=1}^{3} X^{(j)}\beta^{(j)} + \sigma\epsilon \tag{59}$$

where $\beta^{(j)} = (1, 2, 3, 4, 5, 0, ..., 0)$ for $j = 1, 2, 3$ and $\epsilon \sim N(0, 1)$. 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-based joint optimization searched over a 10×10 grid from 1e-5 to $\max(\{||X^{(j)T}y||_2\}_{j=1,...,m})$.

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.

$$\begin{gathered} x_i = Z_1 + \delta_i \text{ for } i = 1, 2, 3 \\ x_i = Z_2 + \delta_i \text{ for } i = 4, 5, 6 \\ x_i \sim N(0, 1) \text{ for } i = 7, ..., 20 \end{gathered} \tag{60}$$

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

$$y = X\beta + \kappa g(z) + \sigma\epsilon \tag{61}$$

where $\beta = (1, 1, 1, 1, 1, 1, 0, ..., 0)$ and $g(z) = (2 - z)\sin(20z^4)$. Constants κ and σ were chosen such that the linear to nonlinear ratio $\frac{||X\beta||_2}{||g(Z)||_2}$ 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-based joint optimization searched over a 10×10 grid from 1e-6 to 10.

3.4 Discussion of results

As shown in Table 2, the descent-based and grid-based joint optimization have the same average performance in all three regression examples. So the two methods have the same performance for simple regularization methods.

We also note that descent-based joint optimization is slightly faster than grid-based optimization. However, for regularization methods with only one or two penalty parameters, we shouldn't expect huge efficiency gains from descent-based joint optimization.

4 Results: regularizations with more than two penalties

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 can find better models in a more complex model space and still run efficiently.

We experimented with generalizations of the simple regressions from the previous section. That is, we try an "unpooled" version of sparse group lasso in which each covariate group has its own regularization parameter. For the APLM example, we add a ridge penalty as a third regularization term. Details regarding the joint optimization formulations and gradient derivations for these generalized regression models are in the Appendix.

We used gradient descent to tune the parameters for these more complex regularization methods. For a baseline reference, we tuned the parameters for the original two-parameter regression using grid-based joint optimization. We compared the accuracies of these models on a separate test set. The results show that gradient descent fit models for the generalized regressions that achieved lower test error. In addition, the method remained computationally tractable, even when tuning over a hundred regularization parameters.

4.1 Unpooled sparse group lasso

We first test an "unpooled" version of sparse group lasso where the group lasso penalty parameter is replaced with individual penalty parameters for each covariate group as follows:

$$\frac{1}{2n}\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\|_2^2 \sum_{m=1}^{M} \lambda_1^{(m)} \|\boldsymbol{\theta}^{(m)}\|_2 + \lambda_2\|\boldsymbol{\theta}\|_1 \tag{62}$$

Consequently, the number of penalty parameters is increased from two to $M+1$, where M is the number of groups. The additional flexibility allows covariate and covariate group effects to be set to zero according to different thresholds. Therefore, this generalized regression 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 in Section 3.2. We used the same grid for grid-based joint optimization. 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 $1e\text{-}2 \times \mathbf{1}^\top$.

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

Furthermore, gradient descent was significantly faster in all settings. The runtimes show that gradient descent does not grow with the number of regularization parameters.

4.2 Additive partially linear models with three regularization parameters

We generalized the APLM criterion in (49) by using the elastic net instead of the lasso penalty, as follows:

n=60, p=300, g=3, M=30				
	β Error	% correct nonzero β	Test Error	Runtime (sec)
Gridsearch (baseline)	1.13	10.70	0.04	15.81
Gradient Descent	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)
Gridsearch (baseline)	7.79	9.63	0.28	148.64
Gradient Descent	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)
Gridsearch (baseline)	2.20	10.69	0.080	162.14
Gradient Descent	0.06	15.34	0.002	48.63

Table 3. Unpooled sparse group lasso

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

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

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

Table 4. additive partially linear Model

$$\frac{1}{2}\left\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\beta} - \boldsymbol{I}_T\boldsymbol{\theta}\right\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\lambda_2\|\boldsymbol{\beta}\|_2^2 + \frac{1}{2}\lambda_3\|D^{(2)}\boldsymbol{\theta}\|_2^2 \tag{63}$$

Since the elastic net tends to perform better when some of the predictors are correlated, we hypothesize that this generalized APLM will fit better models when the linear predictors are to be correlated. Given the scalability of descent-based joint optimization, we were able to test this.

We used the same simulation settings as those in Section 3.3, in which two groups of three covariates were correlated. Since there are three regularization parameters in this model, gradient descent was initialized at $\boldsymbol{\lambda} = 10^i \times \mathbf{1}_3^\top$ for $i = -4, ..., 1$. We experimented with three nonlinear functions $g : \mathbb{R} \mapsto \mathbb{R}$ of varying levels of smoothness, which are given in Table 4.

In addition to judging models by their performance on the test set, we measured the error of the fitted linear effects and the nonparametric estimates. These correspond to the β error $(||\beta - \hat{\beta}||_2^2)$ and θ error $(||g(z) - \theta||_2^2)$, 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 was slightly longer than tuning two parameters with the grid-based method. Nonetheless, the runtime remained reasonable.

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

Table 5. Ulcerative Colitis Data: SGL = sparse group lasso, variance in parentheses

5 Application to Biological Data

We have also applied descent-based joint optimization to a real data example. We have chosen the problem of finding genes from gene pathways that drive Crohn's Disease and Ulcerative Colitis, which was addressed by Simon et al. [2013] using the sparse group lasso. As a comparison, we tackle the same problem but use the un-pooled sparse group lasso and tune its regularization parameters 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 health 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. 5-fold cross validation was used to fit models. The penalty parameters in un-pooled sparse group lasso were initialized at $0.5 \times \mathbf{1}^\top$. For sparse group lasso, models were fit over a 5×5 grid of parameter values from $1e$-4 to 5. Table 5 presents the average results from repeating this process ten times.

The results show that unpooled sparse group lasso achieved a slightly higher classification rate than sparse group lasso. Interestingly, unpooled sparse group lasso finds solutions that are significantly more sparse than sparse group lasso – on average, 9 genesets were identified, as opposed to 38. In addition, the number of genesets identified by unpooled sparse group lasso has significantly lower variance; from the ten runs, sparse group lasso returned 2 to 73 genesets, whereas un-pooled sparse group lasso returned 8 to 12 genesets. These results suggest that un-pooling the penalty parameters in sparse group lasso could potentially improve interpretability and stability.

Finally, we note that tuning the 327 regularization parameters in unpooled sparse group lasso using gradient descent was computationally tractable. In fact, it was slightly faster than tuning the two regularization parameters in sparse group lasso using grid-based joint optimization.

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-based methods. Furthermore, the scalability of this approach allows us to test new regularization methods with many regularization parameters. 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.

7 Appendix

7.1 Proof of Theorem 1

Proof. We will show that for a given $\boldsymbol{\lambda}_0$ that satisfies the given conditions, the validation loss is continuously differentiable within some neighborhood of $\boldsymbol{\lambda}_0$. It then follows that if the theorem conditions hold true for almost every $\boldsymbol{\lambda}$, then the validation loss is continuously differentiable with respect to λ at almost every λ.

Suppose the theorem conditions are satisfied at $\boldsymbol{\lambda}_0$. Let $\boldsymbol{B}'$ be an orthonormal set of basis vectors that span the differentiable space $\Omega^{L_T}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0), \boldsymbol{\lambda}_0)$ with the subset of vectors $\boldsymbol{B}$ that span the model parameter space.

Let $\tilde{L}_T(\boldsymbol{\theta}, \boldsymbol{\lambda})$ be the gradient of $L_T(\cdot, \boldsymbol{\lambda})$ at $\boldsymbol{\theta}$ with respect to the basis $\boldsymbol{B}$:

$$\tilde{L}_T(\boldsymbol{\theta}, \boldsymbol{\lambda}) =_{\boldsymbol{B}} \nabla L_T(\cdot, \boldsymbol{\lambda})|_{\boldsymbol{\theta}} \tag{64}$$

Since $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0)$ is the minimizer of the training loss, the gradient of $L_T(\cdot, \boldsymbol{\lambda}_0)$ with respect to the basis $\boldsymbol{B}$ must be zero at $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0)$:

$${}_{\boldsymbol{B}}\nabla L_T(\cdot, \boldsymbol{\lambda}_0)|_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0)} = \tilde{L}_T(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0), \boldsymbol{\lambda}_0) = 0 \tag{65}$$

From our assumptions, we know that there exists a neighborhood W containing $\boldsymbol{\lambda}_0$ such that $\tilde{L}_T$ is continuously differentiable along directions in the differentiable space $\Omega^{L_T}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0), \boldsymbol{\lambda}_0)$. Also, the Jacobian matrix $D\tilde{L}_T(\cdot, \boldsymbol{\lambda}_0)|_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0)}$ with respect to basis $\boldsymbol{B}$ is nonsingular. Therefore, by the implicit function theorem, there exist open sets $U \subseteq W$ containing $\boldsymbol{\lambda}_0$ and V containing $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0)$ and a continuously differentiable function $\gamma : U \to V$ such that for every $\boldsymbol{\lambda} \in U$, we have that

$$\tilde{L}_T(\gamma(\boldsymbol{\lambda}), \boldsymbol{\lambda}) = \nabla_B L_T(\cdot, \boldsymbol{\lambda})|_{\gamma(\boldsymbol{\lambda})} = 0 \tag{66}$$

That is, we know that $\gamma(\boldsymbol{\lambda})$ is a continuously differentiable function that minimizes $L_T(\cdot, \boldsymbol{\lambda})$ in the differentiable space $\Omega^{L_T}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0), \boldsymbol{\lambda}_0)$. Since we assumed that the differentiable space is a local optimality space of $L_T(\cdot, \boldsymbol{\lambda})$ in the neighborhood W, then for every $\boldsymbol{\lambda} \in U$,

$$\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta} \in \Omega^{L_T}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}_0), \boldsymbol{\lambda}_0)} L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \gamma(\boldsymbol{\lambda}) \tag{67}$$

Therefore, we have shown that if $\boldsymbol{\lambda}_0$ satisfies the assumptions given in the theorem, the fitted model parameters $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is a continuously differentiable function within a neighborhood of $\boldsymbol{\lambda}_0$. We can then apply the chain rule to get the gradient of the validation loss. □

7.2 Gradient Derivations

7.2.1 Unpooled Sparse Group Lasso

The joint optimization formulation of the unpooled sparse group lasso is

$$\begin{array}{c} \min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \frac{1}{2n} \left\| \boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{where } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2n} \|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}\|_2^2 + \sum_{m=1}^M \lambda_1^{(m)} \|\boldsymbol{\theta}^{(m)}\|_2 + \lambda_2 \|\boldsymbol{\theta}\|_1 + \frac{1}{2}\epsilon \|\boldsymbol{\theta}\|_2^2 \end{array} \tag{68}$$

Let $I(\boldsymbol{\lambda}) = \{i | \hat{\theta}_i(\boldsymbol{\lambda}) \neq 0\}$. With similar reasoning in Section 2.4.3, the differentiable space for this problem is $span(\boldsymbol{I}_{I(\boldsymbol{\lambda})})$. All three conditions of Theorem 1 are satisfied. We note that the Hessian in this problem is

$$\frac{1}{n} \boldsymbol{X}_{T,I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} + \boldsymbol{B}(\boldsymbol{\lambda}) + \epsilon \boldsymbol{I} \tag{69}$$

where $\boldsymbol{B}(\boldsymbol{\lambda})$ is the block diagonal matrix with components $m = 1, 2, ..., M$

$$\frac{\lambda_1^{(m)}}{||\boldsymbol{\theta}^{(m)}||_2} \left(\boldsymbol{I} - \frac{1}{||\boldsymbol{\theta}^{(m)}||_2^2} \boldsymbol{\theta}^{(m)} \boldsymbol{\theta}^{(m)\top} \right) \tag{70}$$

from top left to bottom right. This is positive definite for any $\epsilon > 0$.

To find the gradient, the locally equivalent joint optimization with a smooth training criterion is

$$\begin{array}{c} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2_+} \frac{1}{2n} \left\| \boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n} \left\| \boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})}\boldsymbol{\beta} \right\|_2^2 + \sum_{m=1}^M \lambda_1^{(m)} \|\boldsymbol{\beta}^{(m)}\|_2 + \lambda_2\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\epsilon\|\boldsymbol{\beta}\|_2^2 \end{array} \tag{71}$$

Implicit differentiation of the gradient condition with respect to the regularization parameters gives us

$$\begin{aligned} \frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) &= \begin{bmatrix} \frac{\partial}{\partial\boldsymbol{\lambda}_1^{(1)}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) & \cdots & \frac{\partial}{\partial\boldsymbol{\lambda}_1^{(M)}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial\boldsymbol{\lambda}_2}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \end{bmatrix} \\ &= -\left(\frac{1}{n}\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} + \boldsymbol{B}(\boldsymbol{\lambda}) + \epsilon\boldsymbol{I}\right)^{-1} \begin{bmatrix} C(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) & sgn(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \end{bmatrix} \end{aligned} \tag{72}$$

where $C(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}))$ has columns $m = 1, 2..., M$

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ \frac{\hat{\boldsymbol{\beta}}^{(m)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(m)}(\boldsymbol{\lambda})||_2} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{73}$$

By the chain rule, we get that the gradient of the validation error is

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y}_V, X_V\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) = \frac{1}{n}\left(X_{V,I(\boldsymbol{\lambda})}\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right)^\top (\boldsymbol{y}_V - X_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \tag{74}$$

7.2.2 Additive Partially Linear Model with three penalties

The joint optimization formulation of the additive partially linear model with the elastic net penalty for the linear model $\boldsymbol{\beta}$ and the H-P filter for the nonparametric estimates $\boldsymbol{\theta}$ is

$$\begin{array}{c} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2_+} \frac{1}{2} \left\| \boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - (\boldsymbol{I} - \boldsymbol{I}_T)\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta},\boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\beta} - \boldsymbol{I}_T\boldsymbol{\theta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\lambda_2\|\boldsymbol{\beta}\|_2^2 + \frac{1}{2}\lambda_3\|D(\boldsymbol{z})\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\epsilon\|\boldsymbol{\theta}\|_2^2 \end{array} \tag{75}$$

The differentiable space is exactly the same as that given in Section 2.4.5. Also, all three conditions of Theorem 1 are satisfied. Note that the Hessian of the training criterion with respect to the basis in 53 is

$$H = \begin{bmatrix} \boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} + \lambda_2\boldsymbol{I} & \boldsymbol{I}_{I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_T^\top \boldsymbol{I}_T \\ \boldsymbol{I}_T^\top \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})} & \boldsymbol{I}_T^\top \boldsymbol{I}_T + \lambda_3\boldsymbol{D}(\boldsymbol{z})^\top\boldsymbol{D}(\boldsymbol{z}) + \epsilon\boldsymbol{I} \end{bmatrix} \tag{76}$$

To find the gradient, we first consider the locally equivalent joint optimization problem with a smooth training criterion:

$$\begin{array}{c} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2_+} \frac{1}{2} \left\| \boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) - (\boldsymbol{I} - \boldsymbol{I}_T)\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{where } \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\eta},\boldsymbol{\theta}} \frac{1}{2} \left\| \boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})}\boldsymbol{\eta} - \boldsymbol{I}_T\boldsymbol{\theta} \right\|_2^2 + \lambda_1\|\boldsymbol{\eta}\|_1 + \frac{1}{2}\lambda_2\|\boldsymbol{\eta}\|_2^2 + \frac{1}{2}\lambda_3\|\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\epsilon\|\boldsymbol{\theta}\|_2^2 \end{array} \tag{77}$$

After implicit differentiation of the gradient condition with respect to the regularization parameters, we get that

$$\begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \lambda_1} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_3} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_3} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \lambda_1} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_2} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) & \frac{\partial}{\partial \lambda_3} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{bmatrix} = -H^{-1} \begin{bmatrix} sgn(\hat{\boldsymbol{\eta}}(\boldsymbol{\lambda})) & \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{D}(\boldsymbol{z})^\top \boldsymbol{D}(\boldsymbol{z}) \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{bmatrix} \tag{78}$$

We then apply the chain rule to get the gradient direction of the validation loss with respect to $\boldsymbol{\lambda}$

$$\nabla_{\boldsymbol{\lambda}} L_V(\boldsymbol{\lambda}) = -\left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) + (\boldsymbol{I} - \boldsymbol{I}_T) \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}) - (\boldsymbol{I} - \boldsymbol{I}_T) \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right) \tag{79}$$

7.3 Backtracking Line Search

Let the criterion function be $L : \mathbb{R}^n \to \mathbb{R}$, the current point x, and a descent direction Δx. Backtracking line search uses a heuristic for finding a step size $t \in (0, 1]$ such that the value of the criterion is minimized. The method depends on constants $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$.

Algorithm 4 Backtracking Line Search

```
Initialize $t = 1$.
while $L(x + t\Delta x) > L(x) + \alpha t \nabla L(x)^T \Delta x$ do
    Update $t := \beta t$
end while
```

7.4 Joint Optimization with Accelerated Gradient Descent and Adaptive Restarts

Algorithm 5 Joint Optimization with Accelerated Gradient Descent and Adaptive Restarts

```
Initialize $\boldsymbol{\lambda}^{(0)}$.
while stopping criteria is not reached do
  for each iteration $k = 0, 1, ...$ do
    Solve for $\hat{\theta}(\boldsymbol{\lambda}^{(k)}) = \arg\min_{\theta \in \mathbb{R}^p} L_T(\theta, \boldsymbol{\lambda}^{(k)})$.
    Construct matrix $U^{(k)}$, an orthonormal basis of $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}\left(\hat{\theta}\left(\boldsymbol{\lambda}^{(k)}\right)\right)$.
    Define the locally equivalent joint optimization problem

      $$\begin{array}{c} \min_{\boldsymbol{\lambda}} L(y_V, f_{U^{(k)}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(X_V)) \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} L(y_T, f_{U^{(k)}\boldsymbol{\beta}}(X_T)) + \sum_{i=1}^{J} \lambda_i P_i(U^{(k)}\boldsymbol{\beta}) \end{array} \quad (80)$$

    Calculate $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\beta}(\boldsymbol{\lambda})|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}}$ where

      $$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\left[ {}_{U^{(k)}}\nabla^2 \left( L(\boldsymbol{y_T}, f_{U^{(k)}\boldsymbol{\beta}}(\boldsymbol{X_T})) + \sum_{i=1}^{J} \lambda_i P_i(U^{(k)}\boldsymbol{\beta}) \right) \Bigg|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right]^{-1} \left[ {}_{U^{(k)}}\nabla P(U^{(k)}\boldsymbol{\beta})|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right] \quad (81)$$

    with ${}_{U^{(k)}}\nabla^2$ and ${}_{U^{(k)}}\nabla$ are as defined in (15).
    Calculate the gradient $\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V}))|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}}$ where

      $$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V})) = \left[ U^{(k)} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right]^{\top} \left[ {}_{U^{(k)}}\nabla L(\boldsymbol{y_V}, f_{U^{(k)}\boldsymbol{\beta}}(\boldsymbol{X_V}))\big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})} \right] \quad (82)$$

    Perform Neterov's update with step size $t^{(k)}$:

      $$\begin{array}{lcl} \boldsymbol{\eta} & := & \boldsymbol{\lambda}^{(k)} + \frac{k-1}{k+2}\left(\boldsymbol{\lambda}^{(k)} - \boldsymbol{\lambda}^{(k-1)}\right) \\ \boldsymbol{\lambda}^{(k+1)} & := & \boldsymbol{\eta} - t^{(k)}\, \nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V})\right)\Big|_{\boldsymbol{\lambda}=\boldsymbol{\eta}} \end{array} \quad (83)$$

    if the stopping criteria is reached or

      $$L\left(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda}^{(k+1)})}(\boldsymbol{X}_V)\right) > L\left(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda}^{(k)})}(\boldsymbol{X}_V)\right), \quad (84)$$

     then
      set $\boldsymbol{\lambda}^{(0)} := \boldsymbol{\lambda}^{(k)}$ and break
    end if
  end for
end while
return $\boldsymbol{\lambda}^{(0)}$ and $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(0)})$
```

References

S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

M. E. Burczynski, R. L. Peterson, N. C. Twine, K. A. Zuberek, B. J. Brodeur, L. Casciotti, V. Maganti,

P. S. Reddy, A. Strahs, F. Immermann, et al. 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, 2006.

S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 2016. URL `http://stanford.edu/~boyd/papers/pdf/cvxpy_paper.pdf`. To appear.

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

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

A. Lorbert and P. J. Ramadge. Descent methods for tuning parameter refinement. In *International Conference on Artificial Intelligence and Statistics*, pages 469–476, 2010.

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

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

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

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

N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, et al. 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, 2005.

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

A. Tsybakov. *Introduction to Nonparametric Estimation.* Springer Series in Statistics. Springer, 2008. ISBN 9780387790527. URL `https://books.google.com/books?id=mwB8rUBsbqoC`.

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

H. Zou and T. Hastie. Regression shrinkage and selection via the elastic net, with applications to microarrays. *Journal of the Royal Statistical Society: Series B. v67*, pages 301–320, 2003.