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 and a response measured on n observations. Let X denote the $n \times p$ design matrix and y the response vector. 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 2007), which combines the group lasso and lasso penalties. The general form of these regression problems is:

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

where L is the loss function, $\{P_i\}_{i=1,...,J}$ are the penalty functions, and $\{\lambda_i\}_{i=1,...,J}$ 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 that minimize the validation error, as formulated in the following optimization problem:

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

The simplest approach to solving the problem above 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 train/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.

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 joint optimization to both linear and nonlinear regressions. In particular this paper focuses on three examples that demonstrate the wide applicability of our method: elastic net, sparse group lasso, and additive partial 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

Suppose that we have n observations on each of which we have measured p features and a response. Let $\boldsymbol{y}$ be the n response vector and $\boldsymbol{X}$ be the $n \times p$ design matrix.

We are interested in finding the optimal model f in some model class F that minimizes a penalized criterion of the form:

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

where $L : \mathbb{R}^{2n} \to \mathbb{R}$ is the loss function, $P_i : \mathbb{R}^{m_i} \to \mathbb{R}$ is the ith penalty function, and λ_i are the corresponding penalty parameters. In this paper, we will assume that L and P_i are convex functions. Since the optimization problem in (3) only requires fitted values at the points in the design matrix $\boldsymbol{X}$ and the penalty functions take in finite-dimensional inputs, the optimization problem reduces to optimizing over some finite-dimensional space Θ. If F is the space of parametric models, then Θ would be the space of possible model parameters. If F is the space of nonparametric models, then Θ is an $n + k$-dimensional space where the first n dimensions are the fitted values for the points in the design matrix and the last k dimensions are any additional parameters required for specifying the nonparametric model. We can therefore rewrite the optimization problem in the following form instead:

$$\min_{\theta \in \Theta} L(\boldsymbol{y}, f_\theta(\boldsymbol{X})) + \sum_{i=1}^{J} \lambda_i P_i(\theta) \tag{4}$$

Suppose that we use a training/validation split to select penalty parameters $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_J)^T$. 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 choose our

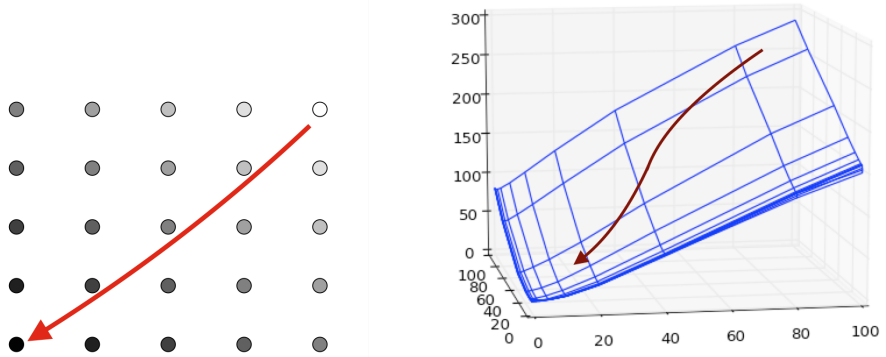

Figure 1. Left: darker points mean lower validation loss. Descent-based optimization descends in the most direct path towards the point producing the lowest validation loss. Right: The 3D version. We can tune regularization parameters using a grid search... or just descend opposite of the gradient.

penalty parameters to minimize the validation error by solving the following optimization problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{\lambda}\in\mathbb{R}^k} L(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ &\text{where } \hat{\theta}(\boldsymbol{\lambda}) = \arg\min_{\theta\in\Theta} L(\boldsymbol{y}_T, f_\theta(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\theta) \end{aligned} \tag{5}$$

(5) 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, (5) 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 (5). 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 an active 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. The major benefit of using our method is that it opens up the possibility of using regularization methods that combine multiple 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

$$L_T(\theta, \boldsymbol{\lambda}) \equiv L(\boldsymbol{y}_T, f_\theta(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\theta) \tag{6}$$

First we consider the simple case where $L_T(\theta, \boldsymbol{\lambda})$ 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 solve the joint optimization problem:

Initialize $\boldsymbol{\lambda}^{(0)}$.
For each iteration k:

$$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - \alpha^{(k)} \nabla_{\boldsymbol{\lambda}} L\big(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\big)\big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} \tag{7}$$

Note: $\alpha^{(k)}$ is the step size

Algorithm 1

To calculate the gradient, we first apply the chain rule:

$$\nabla_{\boldsymbol{\lambda}} L\Big(\boldsymbol{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\Big) = \Big[\frac{\partial}{\partial \theta} L(\boldsymbol{y}_V, f_\theta(\boldsymbol{X}_V))\Big|_{\theta=\hat{\theta}(\lambda)}\Big]^T \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda}) \tag{8}$$

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 (6). Since (6) is smooth,

$$\nabla_\theta \Big(L(\boldsymbol{y}_T, f_\theta(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\theta)\Big)\Big|_{\theta=\hat{\theta}(\boldsymbol{\lambda})} = \mathbf{0}. \tag{9}$$

Taking the derivative of both sides of (9) in $\boldsymbol{\lambda}$ and solving for $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda})$, we get:

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda}) = -\left[\nabla_\theta^2 \left(L(\boldsymbol{y}_T, f_\theta(\boldsymbol{X}_T)) + \sum_{i=1}^{J} \lambda_i P_i(\theta)\right)\right]^{-1} \begin{pmatrix} | & | & & | \\ \nabla_\theta P_1(\theta) & \nabla_\theta P_2(\theta) & \ldots & \nabla_\theta P_J(\theta) \\ | & | & & | \end{pmatrix}\Big|_{\theta=\hat{\theta}(\boldsymbol{\lambda})} \tag{10}$$

We can plug this back into (8) to get our gradient. Note that because $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda})$ is defined in terms of $\hat{\theta}(\boldsymbol{\lambda})$, each gradient step requires solving the penalized regression problem on the training data.

2.3 Nonsmooth Training Criterion

When the training criterion in the penalized regression is not smooth, a common approach is to use slower but more general optimization methods. However, for many of these problems, the solution $\hat{\theta}(\boldsymbol{\lambda})$ is 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 (8) and apply gradient descent in practice. To provide a general approach to solving such problems, we discuss two primary difficulties: the characterization of problems for which $\hat{\theta}(\boldsymbol{\lambda})$ is almost everywhere smooth and the actual calculations for $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda})$.

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 the set of vectors $\Omega^L(\eta)$ such that the directional derivative of L at η along $u \in \Omega^L(\eta)$ exists.*

$$\Omega^L(\eta) = \Big\{\delta \big| \lim_{\epsilon \to 0} \frac{L(\eta + \epsilon\delta) - L(\eta)}{\epsilon} \text{ exists } \Big\} \tag{11}$$

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

$$\arg\min_{\theta} L(\theta, \lambda) = \arg\min_{\theta \in S} L(\theta, \lambda) \tag{12}$$

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

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

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 $\boldsymbol{\lambda}$, the Hessian of $L_T(\cdot, \boldsymbol{\lambda})$ at $\hat{\theta}(\boldsymbol{\lambda})$ with respect to a basis of $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))$ is invertible.*

Putting all of these together we get the following theorem which shows that the gradient exists almost everywhere; and gives a recipe for calculating it.

Theorem 1. *Suppose our optimization problem is of the form in (5), with $L_T(\theta, \boldsymbol{\lambda})$ defined as in (6).*

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

Then the validation loss $L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V}))$ is continuously differentiable almost everywhere with respect to $\boldsymbol{\lambda}$. Furthermore, the gradient of $L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V}))$, where it is defined, is

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V})) = \Big[\frac{\partial}{\partial \theta} L(\boldsymbol{y_V}, f_{\theta}(\boldsymbol{X_V}))|_{\theta=\tilde{\theta}(\lambda)}\Big]^T \frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\theta}(\boldsymbol{\lambda}) \tag{13}$$

where

$$\tilde{\theta}(\boldsymbol{\lambda}) = \arg \min_{\theta \in \Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))} L_T(\theta, \boldsymbol{\lambda}) \tag{14}$$

We can therefore construct a gradient descent procedure based on the key idea in (14). At each iteration, we define a smooth, locally equivalent function $\tilde{\theta}(\boldsymbol{\lambda})$ and use $\frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\theta}(\boldsymbol{\lambda})$ to calculate the gradient. The gradient descent procedure is given in more detail below.

Initialize $\boldsymbol{\lambda}^{(0)}$.
For each iteration k until stopping criteria is reached:

Solve for $\hat{\theta}(\boldsymbol{\lambda}^{(k)}) = \arg\min_{\theta \in \Theta} L_T(\theta, \boldsymbol{\lambda}^{(k)})$.

Let $\tilde{\theta}(\boldsymbol{\lambda}^{(k)})$ be the optimal model parameters restricted to the space $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}\left(\hat{\theta}\left(\boldsymbol{\lambda}^{(k)}\right)\right)$.

$$\tilde{\theta}(\boldsymbol{\lambda}^{(k)}) = \arg \min_{\theta \in \Omega^{L_T(\cdot,\boldsymbol{\lambda})}\left(\hat{\theta}\left(\boldsymbol{\lambda}^{(k)}\right)\right)} L_T(\theta, \boldsymbol{\lambda}^{(k)}) \tag{15}$$

Calculate the gradient

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

Perform the gradient update

$$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - \alpha^{(k)} \nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V}))|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}}$$

Algorithm 2

	Differentiable Space
Ridge Regression	$\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) = \mathbb{R}^p$
Elastic Net	$\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) = span(\{e_i \vert \hat{\theta}_i(\boldsymbol{\lambda}) \neq 0\})$
Sparse Group Lasso	$\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) = span(\{e_i \vert \hat{\theta}_i(\boldsymbol{\lambda}) \neq 0\})$
Nesterov's Gradient Descent	$\Omega^{L_T(\cdot,\cdot,\boldsymbol{\lambda})}(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) = \mathbb{R}^{p+n} \setminus span(\{e_i \vert \hat{\beta}_i(\boldsymbol{\lambda}) \neq 0\})$

Table 1. The differentiable space of each example regression problem

2.4 Joint optimization for example regressions

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 and sparse group lasso, in which the training criterion is smooth almost everywhere. Finally, we discuss doing descent-based joint optimization for an additive partial linear model, an example of a semi-parametric regression.

For reference, the differentiable space of each regression example is specified in Table 1.

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:

$$\begin{gathered} \min_{\lambda \in \mathbb{R}} \frac{1}{2}||\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\lambda)||_2^2 \\ \text{where } \hat{\boldsymbol{\theta}}(\lambda) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}||\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}||_2^2 + \frac{1}{2}\lambda||\boldsymbol{\theta}||_2^2 \end{gathered} \tag{17}$$

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

$$\hat{\boldsymbol{\theta}}(\lambda) = (\boldsymbol{X}_T^T \boldsymbol{X}_T + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}_T^T \boldsymbol{y}_T \tag{18}$$

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.

$$\nabla_\lambda L(\boldsymbol{y_V}, f_{\hat{\theta}(\lambda)}(\boldsymbol{X_V})) \quad = \quad (\boldsymbol{X}_V(\boldsymbol{X}_T^T \boldsymbol{X}_T + \lambda \boldsymbol{I})^{-1} \hat{\boldsymbol{\theta}}(\lambda))^T (\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\lambda)) \tag{19}$$

2.4.2 Elastic Net

The elastic net, a linear combination of the lasso and ridge penalties, is an example of a regularization method that is not smooth. We are interested in finding the optimal regularization parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^T$, which is formulated as follows:

$$\begin{gathered} \min_{\boldsymbol{\lambda} \in \mathbb{R}^2} \frac{1}{2}||\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\lambda)||^2 \\ \text{where } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}||\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}||^2 + \lambda_1||\boldsymbol{\theta}||_1 + \frac{1}{2}\lambda_2||\boldsymbol{\theta}||_2^2 \end{gathered} \tag{20}$$

This joint optimization problem satisfies all three conditions in theorem 1. For the first condition, note that $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))$ is the space spanned by the nonzero elements of $\hat{\boldsymbol{\theta}}(\lambda)$. Furthermore, for the elastic net penalty, it is known that the nonzero indices of the fitted regression coefficients stay locally constant for almost every $\boldsymbol{\lambda}$. Therefore, $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))$ is a local optimality space. The second condition is also satisfied since the ℓ_1 penalty is smooth when restricted to $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))$. Finally, to check the third condition, let $\tilde{\boldsymbol{X}}(\boldsymbol{\lambda})$ be the rows of $\boldsymbol{X}$ projected onto $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))$. The Hessian of $L_T(\cdot, \boldsymbol{\lambda})$ with respect to the standard basis of $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}(\boldsymbol{\lambda}))$ is $\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T \tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda}) + \lambda_2 \boldsymbol{I}$, which is always invertible.

To calculate the gradient, we consider the minimization problem restricted to $\Omega^{L_T(\cdot,\lambda)}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$, the space spanned by the non-zero elements of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$:

$$\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg \min_{\{\theta_i | \hat{\theta}_i(\boldsymbol{\lambda}) \neq 0\}} ||\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}||_2^2 + \lambda_1||\boldsymbol{\theta}||_1 + \lambda_2||\boldsymbol{\theta}||_2^2 \tag{21}$$

From (10), we get that

$$\frac{\partial}{\partial \lambda_1}\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = (\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda}) + \lambda_2\boldsymbol{I})^{-1} sgn\left(\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})\right) \tag{22}$$

$$\frac{\partial}{\partial \lambda_2}\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = (\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda}) + \lambda_2\boldsymbol{I})^{-1}\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \tag{23}$$

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

$$\nabla_\lambda L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\lambda)}(\boldsymbol{X_V})) = \begin{bmatrix} sgn(\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})) \\ \tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{bmatrix}^T (\tilde{\boldsymbol{X}}_V(\boldsymbol{\lambda})^T(\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda}) + \lambda_2\boldsymbol{I})^{-1})^T(\boldsymbol{y}_V - \tilde{\boldsymbol{X}}_V(\boldsymbol{\lambda})\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})) \tag{24}$$

2.4.3 Sparse Group Lasso

The sparse group lasso combines the $||\cdot||_2$ and $||\cdot||_1$ penalties, both of which are not smooth. This regularization method is particularly well-suited to modeling problems in which the grouped covariates are thought to have a sparse effect on a group level and within groups.

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

$$\begin{gathered} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2} \frac{1}{2n}\left|\left|\boldsymbol{y}_V - X_V\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\right|\right|_2^2 \\ \text{where } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2n}\left|\left|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\right|\right|_2^2 + \lambda_1 \sum_{m=1}^{M} ||\boldsymbol{\theta}^{(m)}||_2 + \lambda_2||\boldsymbol{\theta}||_1 \end{gathered} \tag{25}$$

First, we note that this problem satisfies the three conditions given in theorem 1. The reasoning is very similar to that in the elastic net example so we do not repeat them here. Since $||\cdot||_2$ and $||\cdot||_1$ are not differentiable along the same directions at every point, $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}\,(\boldsymbol{\lambda}))$ is the space spanned by the nonzero elements of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$.

To calculate the gradient, we solve for $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ and define the locally equivalent function $\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})$:

$$\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg \min_{\{\theta_i|\hat{\theta}_i(\boldsymbol{\lambda})\neq 0\}} \frac{1}{2n}\left|\left|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\right|\right|_2^2 + \lambda_1 \sum_{m=1}^{M} ||\boldsymbol{\theta}^{(m)}||_2 + \lambda_2||\boldsymbol{\theta}||_1 \tag{26}$$

Taking the gradient of the expression above with respect to the basis of $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\theta}\,(\boldsymbol{\lambda}))$ gives us equation (9) in the context of this problem:

$$-\frac{1}{n}\tilde{\boldsymbol{X}}_T^T(\boldsymbol{y}_T - \tilde{\boldsymbol{X}}_T\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})) + \lambda_1 \begin{bmatrix} \frac{\tilde{\boldsymbol{\theta}}^{(1)}}{||\tilde{\boldsymbol{\theta}}^{(1)}||_2} \\ ... \\ \frac{\tilde{\boldsymbol{\theta}}^{(M)}}{||\tilde{\boldsymbol{\theta}}^{(M)}||_2} \end{bmatrix} + \lambda_2 sgn(\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})) = 0 \tag{27}$$

From (10) and the chain rule, we get the gradient of the validation loss:

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X_V})) = \frac{1}{n}\begin{bmatrix} \frac{\tilde{\boldsymbol{\theta}}^{(1)}}{||\tilde{\boldsymbol{\theta}}^{(1)}||_2} \\ ... \\ \frac{\tilde{\boldsymbol{\theta}}^{(M)}}{||\tilde{\boldsymbol{\theta}}^{(M)}||_2} \\ sgn(\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})) \end{bmatrix}^T \left(\tilde{\boldsymbol{X}}_V(\boldsymbol{\lambda})(\frac{1}{n}\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda}) - \lambda_1\boldsymbol{M}_1 + \lambda_1\boldsymbol{M}_2)^{-1}\right)^T (\boldsymbol{y}_V - \tilde{\boldsymbol{X}}_V(\boldsymbol{\lambda})\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})) \tag{28}$$

where $\boldsymbol{M}_1$ is the block diagonal matrix with components $||\tilde{\boldsymbol{\theta}}^{(m)}(\boldsymbol{\lambda})||_2^{-3}\tilde{\boldsymbol{\theta}}^{(m)}(\boldsymbol{\lambda})\tilde{\boldsymbol{\theta}}^{(m)}(\boldsymbol{\lambda})^T$ and $\boldsymbol{M}_2$ is the block diagonal matrix with components $||\tilde{\boldsymbol{\theta}}^{(k)}(\boldsymbol{\lambda})||_2^{-1}\boldsymbol{I}$ for $m = 1, ..., M$ from top left to bottom right.

2.4.4 Additive Partial Linear Models

Finally, we present an example of a penalized, semi-parametric regression: an additive partial linear model (APLM) combined with the lasso and trend-filtering penalties. This regression method models the response as the sum of linear and nonlinear components and selects model fits that have sparse linear effects and smooth nonparametric estimates.

The setup for this semi-parametric problem is slightly different. Let $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ be the design matrix with p linear predictors, $\boldsymbol{z} \in \mathbb{R}^n$ be the design matrix with the nonlinear predictor, and $\boldsymbol{y} \in \mathbb{R}^n$ be the response variable. We assume that the observations are ordered by increasing z_i. Let $\boldsymbol{M}$ a diagonal matrix with elements randomly assigned to 1 or 0. $\boldsymbol{M}$ partitions the data into the training set $\boldsymbol{X}_T = \boldsymbol{M}\boldsymbol{X}$ and the validation set $\boldsymbol{X}_V = (\boldsymbol{I} - \boldsymbol{M})\boldsymbol{X}$.

The response y_i in this APLM is modeled as the sum of a linear function of x_i and a nonlinear function of z_i

$$y_i = \boldsymbol{x_i}^T \boldsymbol{\beta} + g(z_i) + \epsilon \tag{29}$$

where $\boldsymbol{\beta}$ is an unknown vector and g is an unknown nonlinear function. We want to solve for the optimal $\boldsymbol{\beta}$ and the trend-filtering estimates $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)$ for $(g(z_1), ..., g(z_n))^T$. Since these values depend on the regularization parameters, we are ultimately interested in finding the optimal $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^T$. The joint optimization problem is as follows:

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

The third component in the training criterion is the trend-filtering penalty $||\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}||_2^2$. We use the second-order $||\cdot||_2^2$ trend-filtering penalty in this example, which penalizes the norm of the second-order discrete derivatives over the input points. The second-order difference matrix $\boldsymbol{D}(\boldsymbol{z})$ is defined as

$$\boldsymbol{D}(\boldsymbol{z}) = \boldsymbol{D}^{(1)} \cdot \operatorname{diag}\Big(\frac{1}{z_2 - z_1}, \frac{1}{z_3 - z_2}, ..., \frac{1}{z_n - z_{n-1}}, 0\Big) \cdot \boldsymbol{D}^{(1)} \tag{31}$$

where

$$\boldsymbol{D}^{(1)} = \begin{bmatrix} -1 & 1 & 0 & ... & 0 & 0 \\ 0 & -1 & 1 & ... & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & ... & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{32}$$

Note that (30) requires $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ to include the nonparametric estimates for the validation set. We accomplish this by constructing $\boldsymbol{D}(\boldsymbol{z})$ from the nonlinear predictors from the training and validation sets.

We can apply gradient descent by the same reasoning as before, though the differentiable space is slightly different. Let the training criterion be denoted as $L_T(\boldsymbol{\beta}(\boldsymbol{\lambda}), \boldsymbol{\theta}(\boldsymbol{\lambda}), \boldsymbol{\lambda})$. Then $\Omega^{L_T(\cdot,\cdot,\boldsymbol{\lambda})}(\hat{\beta}(\boldsymbol{\lambda}), \hat{\theta}(\boldsymbol{\lambda}))$ is the space spanned by the nonzero elements of $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ and all elements of $\hat{\theta}(\boldsymbol{\lambda})$.

To calculate the gradient, we solve for the $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ and define $\tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ as

$$\tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg \min_{\{\beta_i | \hat{\beta}_i(\boldsymbol{\lambda}) \neq 0\}, \boldsymbol{\theta}} \frac{1}{2} \left|\left| \boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\beta} - \boldsymbol{M}\boldsymbol{\theta} \right|\right|_2^2 + \lambda_1 ||\boldsymbol{\beta}||_1 + \frac{1}{2}\lambda_2 ||\boldsymbol{D}(\boldsymbol{z})\boldsymbol{\theta}||_2^2 \tag{33}$$

Using the same procedure, we see that the gradient of the validation loss is

$$\nabla_\lambda L_V(\boldsymbol{\lambda})_i = -\left(\tilde{\boldsymbol{X}}_V(\boldsymbol{\lambda}) \frac{\partial \tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda})}{\partial \lambda_i} + (\boldsymbol{I} - \boldsymbol{M}) \frac{\partial \tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})}{\partial \lambda_i} \right)^T \left(\boldsymbol{y}_V - \tilde{\boldsymbol{X}}_V(\boldsymbol{\lambda}) \tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - (\boldsymbol{I} - \boldsymbol{M}) \tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right) \text{ for } i = 1, 2$$

where the partial derivatives are

$$\begin{aligned}
\frac{\partial}{\partial\lambda_1}\tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda}) &= -\left(\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda}) - \tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\boldsymbol{M}(\boldsymbol{M}^T\boldsymbol{M}+\lambda_2\boldsymbol{D}(\boldsymbol{z})^T\boldsymbol{D}(\boldsymbol{z}))^{-1}\boldsymbol{M}^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})\right)^{-1} sgn(\tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \\
\frac{\partial}{\partial\lambda_1}\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) &= -\left(\boldsymbol{M}^T\boldsymbol{M}+\lambda_2\boldsymbol{D}(\boldsymbol{z})^T\boldsymbol{D}(\boldsymbol{z})\right)^{-1}\boldsymbol{M}^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})\frac{\partial}{\partial\lambda_1}\tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \\
\frac{\partial}{\partial\lambda_2}\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) &= -\left(\boldsymbol{M}^T\boldsymbol{M} - \boldsymbol{M}^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})(\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda}))^{-1}\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\boldsymbol{M}+\lambda_2\boldsymbol{D}(\boldsymbol{z})^T\boldsymbol{D}(\boldsymbol{z})\right)\boldsymbol{D}(\boldsymbol{z})^T\boldsymbol{D}(\boldsymbol{z})\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \\
\frac{\partial}{\partial\lambda_2}\tilde{\boldsymbol{\beta}}(\boldsymbol{\lambda}) &= -\left(\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})\right)^{-1}\tilde{\boldsymbol{X}}_T(\boldsymbol{\lambda})^T\boldsymbol{M}\frac{\partial}{\partial\lambda_2}\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})
\end{aligned}$$

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.

Grid-based joint optimization was implemented using cross validation over a grid of parameter values. Descent-based joint optimization was implemented in two ways: gradient descent with constant step size and Nesterov's gradient descent with adaptive restarts. For more details on the latter method, refer O'Donaghue and Candes. In brief, Nesterov's gradient descent is an accelerated "descent" method in which step size and direction depends on a momentum that grows at each iteration. With adaptive restarts, the we reset the momentum to zero whenever the criterion starts to increase.

The simulation experiments were on the example regressions elastic net, sparse group lasso, and APLM. Below, we describe the simulation settings, followed by a discussion of the results.

3.1 Elastic net

We generated 30 datasets of 80 training and 20 validation observations with 250 predictors. The predictors were randomly generated from a normal distribution with mean zero, variance one, and pairwise correlation between predictors x_i and x_j as $0.5^{|i-j|}$. The response vector y was generated by

$$y = X\beta + \sigma\epsilon \tag{34}$$

where

$$\beta = (\underbrace{1,...,1}_{\text{size 15}},\underbrace{0,...,0}_{\text{size 235}}) \tag{35}$$

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 used a 10×10 grid from 1e-5 to four times the largest eigenvalue of $X_T^T X_T$.

3.2 Sparse group lasso

We generated 30 datasets of 60 training and 15 validation observations with 1500 covariates. The predictors X were generated from a standard normal distribution. The response y was constructed as

$$y = \sum_{j=1}^{3} X^{(j)}\beta^{(j)} + \sigma\epsilon \tag{36}$$

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. The number of groups m in the criterion was 150.

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

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)

3.3 Additive partial linear models

We generated 30 datasets 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.

$$\begin{aligned} x_i &= Z_1 + \epsilon_i \text{ for } i = 1, 2, 3 \\ x_i &= Z_2 + \epsilon_i \text{ for } i = 4, 5, 6 \\ x_i &\sim N(0, 1) \text{ for } i = 7, ..., 20 \end{aligned} \tag{37}$$

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

$$y = X\beta + \kappa g(z) + \sigma\epsilon \tag{38}$$

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 used a 10×10 grid from 1e-6 to 10.

3.4 Discussion of results

As shown in Table 2, descent-based and grid-based joint optimization methods all had the same performance. The only major difference is in the runtimes. Compared to the grid-based approach, gradient descent with fixed step size was over 40% faster and Nesterov's gradient descent was over 50% faster.

In Figure 2, we show plots that illustrate the variation in performance of the joint optimization. We show one run where the methods can similar values, one where descent-based joint optimization reaches a significantly lower value, and one where grid-based joint optimization reaches a significantly lower value. However, on average, we expect the methods to have the same performance.

4 Results: regularizations with more than two penalties

Our second simulation study shows that descent-based joint optimization can effectively and efficiently solve regressions with more than two regularization parameters.

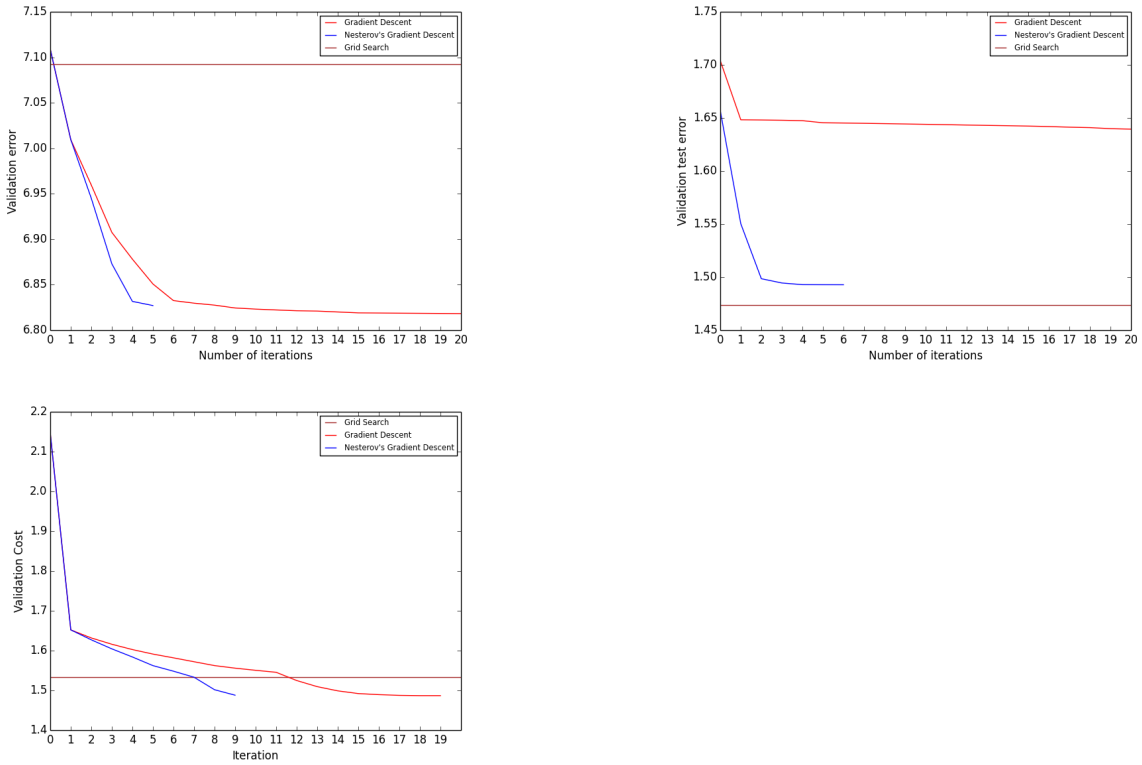

Figure 2. Validation error progression: Top Left - Elastic Net, Top Right - Grouped Lasso, Bottom Left - APLM

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

To do so, we generalized the example regressions above from having two parameters to having many parameters. For the sparse group lasso, we made an "unpooled" version in which the the number of regularization parameters is one plus the number of covariate groups. For the APLM example, we added a ridge penalty to the criterion.

We then compared tuning the two-parameter version using grid-based joint optimization to tuning the many-parameter version using gradient descent. We show that these experiments were computationally tractable and that gradient descent for these generalized regularization methods produced models with lower test error than grid-based joint optimization.

For the joint optimization problem formulations and gradient derivations for these generalized regularizations, please refer to the Appendix.

4.1 Unpooled sparse group lasso

In unpooled sparse group lasso, the group lasso penalty parameter λ_1 is replaced with an individual parameter for each covariate group

$$\sum_{j=1}^{m} \lambda_1^{(j)} ||\beta^{(j)}||_2 + \lambda_2 ||\beta||_1 \tag{39}$$

The number of regularization parameters in (39) is then one plus the number of coefficient groups. This generalized criterion gives models the additional flexibility of having different thresholds for setting covariate and covariate group effects to zero. Therefore, unpooled sparse group lasso can potentially better model 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 previous sparse group lasso simulation study. We used the same grid for grid-based joint optimization. For gradient descent, the $m+1$ regularization parameters were initialized at $1e\text{-}4 \times \mathbf{1}^T$, $1e\text{-}3 \times \mathbf{1}^T$, and $1e\text{-}2 \times \mathbf{1}^T$.

We measured three metrics to assess model performance: test error, β error, which is 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, even in the case with 151 regularization parameters.

$g(z) = 4z^3 - z^2 + 2z$				
	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 Partial Linear Model

4.2 Additive partial linear models with three regularization parameters

We generalized the APLM criterion in (30) by using the elastic net instead of the lasso:

$$\lambda_1||\beta||_1 + \frac{1}{2}\lambda_2||\beta||_2^2 + \frac{1}{2}\lambda_3||D^{(2)}\theta||_2^2 \tag{40}$$

In the simulation, we tested the hypothesis that having the elastic net could better model data with sparse effects and strongly correlated predictors.

We ran experiments on three nonlinear functions of varying levels of smoothness as given in Table 4. The simulation settings were similar to before. The only difference was that gradient descent was initialized at $10^i \times \mathbf{1}^T$ for $i = -4, ..., 1$.

Models were judged by their predictive accuracy. In addition, we measured the error in the fitted linear model and nonparametric estimates with β error=$||\beta - \hat{\beta}||_2^2$ and θ error=$||g(z) - \theta||_2^2$, respectively.

The results show that this generalized APLM criterion was better by all three metrics. The linear model fits improved the most, which is expected since the generalized model added a penalty to the linear coefficients. Surprisingly, the estimation of the nonlinear components also improved slightly, even though no penalty term was added for the nonparametric estimates. The runtime for tuning the three-parameter regularization was slightly longer than the grid-based version, but remained reasonable nonetheless. Perhaps, the most important result is that the runtime of gradient descent is not exponential in the number of penalty parameters.

In Figure 3, we show example model fits from gradient descent and grid-based joint optimization. The plots confirm the hypothesis that the highly-correlated covariates are estimated better using the three-parameter regularization as opposed to the two-parameter one.

5 Application to Real Data

In this section, we apply descent-based joint optimization to real data.

We apply un-pooled sparse group lasso to the real-data application from the sparse group lasso paper Noah et. al. (2007). In this problem, we are interested in finding genes from gene pathways that are driving Crohn's Disease and Ulcerative Colitis.

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 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 2005) and discarded the 2358 genes not found in the gene set.

For each experiment, we randomly shuffled the data and used the first 50 observations for the training set and the remaining 77 for the test set. The experiment was repeated 10 times. 5-fold cross-validation

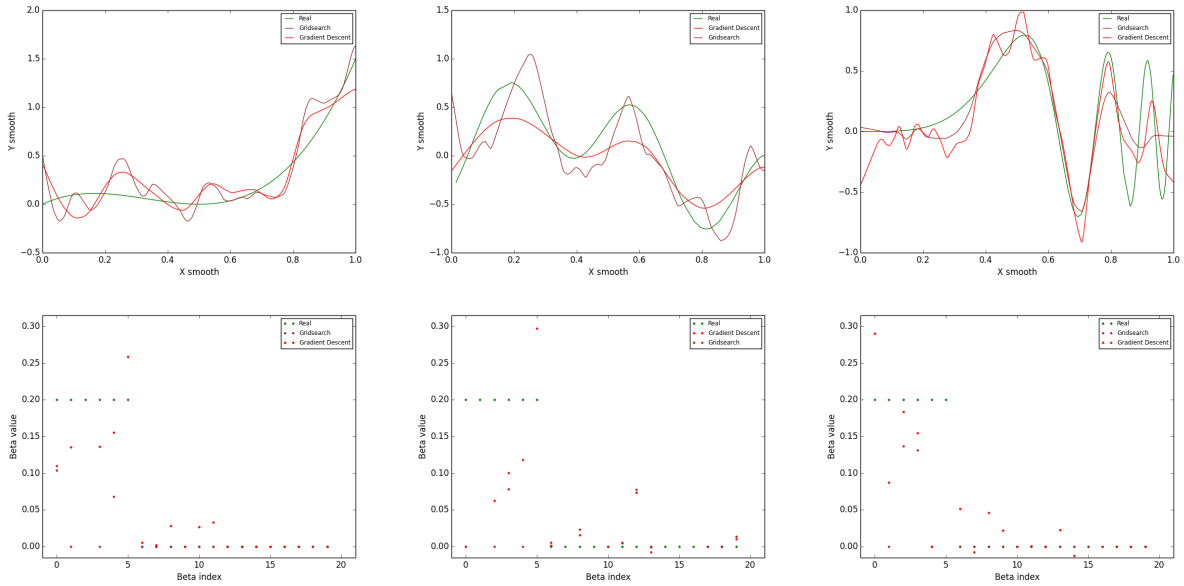

Figure 3. Top: Example fits to the three smooth functions used in the experiments. The green line is the true function. Bottom: Example fits for the linear regression coefficients. The green is the true beta.

	% 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

was used to fit models. The penalty parameters in un-pooled sparse group lasso were initialized at $0.5 \times \mathbf{1}^T$. For sparse group lasso, models were fit over a 5×5 grid of parameter values from 1e-4 to 5.

As shown in Table 5, un-pooled sparse group lasso achieved a slightly lower classification rate compared to sparse group lasso. Interestingly, un-pooled 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 un-pooled sparse group lasso has significantly lower variance. The number of genesets identified by sparse group lasso ranged from 2 to 73, whereas the number of genesets identified by un-pooled sparse group lasso ranged from 8 to 12. These results suggest that un-pooling the penalty parameters in sparse group lasso could potentially improve interpretability and stability. Further research is needed to determine if un-pooled sparse group lasso is a useful regularization method.

Perhaps the most significant result from this section is the fact that this experiment with 327 regularization parameters was tractable. As shown, the runtime of unpooled sparse group lasso was lower than sparse group lasso, even though the latter was only tuning two regularization parameters.

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 have also shown that descent-based joint optimization has similar performance to grid-based methods but is much more efficient.

Due to the scalability of descent-based joint optimization, we were able 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. Further 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.

Appendix

6.1 Proof for Theorem 1

Proof. It is sufficient to show that the theorem holds for a given λ_0. For a given λ_0, let B' be the basis vectors that span the differentiable space $\Omega^{L_T}(\hat{\theta}(\lambda_0), \lambda_0)$. Let B be the subset of the basis vectors that span the model parameter space.

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

$$\tilde{L}(\theta, \lambda) = \nabla_B L_T(\cdot, \lambda)|_\theta \tag{41}$$

Since $\hat{\theta}(\lambda_0)$ is the minimizer of the training loss, the gradient of $L_T(\cdot, \lambda_0)$ with respect to the basis B must be zero at $\hat{\theta}(\lambda_0)$:

$$\nabla_B L_T(\cdot, \lambda_0)|_{\hat{\theta}(\lambda_0)} = \tilde{L}_T(\hat{\theta}(\lambda_0), \lambda_0) = 0 \tag{42}$$

From our assumptions, we know that there exists a neighborhood W such that $\tilde{L}_T$ is continuously differentiable along directions in the differentiable space $\Omega^{L_T}(\hat{\theta}(\lambda_0), \lambda_0)$. Also, the Jacobian matrix $D\tilde{L}_T(\cdot, \lambda_0)|_{\hat{\theta}(\lambda_0)}$ with respect to basis B is nonsingular. Therefore, by the implicit function theorem, there exist open sets U containing λ_0 and V containing $\hat{\theta}(\lambda_0)$ and a continuously differentiable function $\gamma : U \to V$ such that for every $\lambda \in U$, we have that

$$\tilde{L}_T(\gamma(\lambda), \lambda) = \nabla_B L_T(\cdot, \lambda)|_{\gamma(\lambda)} = 0 \tag{43}$$

That is, we know that $\gamma(\lambda)$ is a continuously differentiable function that minimizes $L_T(\cdot, \lambda)$ in the differentiable space $\Omega^{L_T}(\hat{\theta}(\lambda_0), \lambda_0)$. Since we assumed that the differentiable space is a local optimality space of $L_T(\cdot, \lambda)$ in the neighborhood W, then there must exist some neighborhood W' containing λ_0 such that for every $\lambda \in W'$,

$$\hat{\theta}(\lambda) = \arg\min_\theta L_T(\theta, \lambda) = \arg \min_{\theta \in \Omega^{L_T}(\hat{\theta}(\lambda_0), \lambda_0)} L_T(\theta, \lambda) = \gamma(\lambda) \tag{44}$$

Therefore, we have shown that if λ_0 satisfies the assumptions given in the theorem, the fitted model parameters $\hat{\theta}(\lambda)$ is a continuously differentiable function within a neighborhood of λ_0. If the assumptions in the theorem hold true for almost every set of regularization parameters, then $\hat{\theta}(\lambda)$ is continuously differentiable almost everywhere. Furthermore, the Jacobian matrix of the model parameters, where it is defined, is

$$\frac{\partial}{\partial \lambda} \hat{\theta}(\lambda) = \frac{\partial}{\partial \lambda} \dot{\theta}(\lambda) \tag{45}$$

where

$$\dot{\theta}(\lambda) = \arg \min_{\theta \in \Omega^{L_T}(\hat{\theta}(\lambda_0), \lambda_0)} L_T(\theta, \lambda) \tag{46}$$

We can then apply the chain rule to get the gradient of the validation loss. □

6.2 Elastic Net Gradient Derivations

MATHS

6.3 Sparse Group Lasso Derivations

6.3.1 Unpooled Sparse Group Lasso Derivations

The gradient of the validation error with respect to the regularization parameters is:

$$\begin{aligned}\nabla_{\lambda_1^{(k)}} L(\boldsymbol{y}_V, X_V\hat{\beta}) &= \frac{1}{n}\left(X_V(\frac{1}{n}X^TX + M_1 + M_2)^{-1} \begin{bmatrix} 0 \\ \vdots \\ \frac{\hat{\beta}^{(l}}{||\hat{\beta}^{(l}||_2} \\ \vdots \\ 0 \end{bmatrix} \right)^T (\boldsymbol{y}_V - X_V\hat{\beta}) \\ \nabla_{\lambda_2} L(\boldsymbol{y}_V, X_V\hat{\beta}) &= \frac{1}{n}\left((\frac{1}{n}X^TX + M_1 + M_2)^{-1} sign(\hat{\beta}) \right)^T (\boldsymbol{y}_V - X_V\hat{\beta}) \end{aligned} \tag{47}$$

where M_1 is the block diagonal matrix with l components $\frac{\lambda_1^{(k)}}{||\beta^{(k)}||_2^3} \begin{bmatrix} \beta_1^{(k)} & 0 & \\ 0 & \beta_2^{(k)} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} — & \beta^{(k)T} & — \\ — & \beta^{(k)T} & — \\ \vdots & & \end{bmatrix}$ from top left to bottom right and M_2 is the diagonal matrix with $\frac{\lambda_1^{(k)}}{||\beta^{(k)}||_2}$ from top left to bottom right.

6.4 Additive Partial Linear Model Derivations

6.4.1 Additive Partial Linear Model Derivations: 3-penalties

To perform joint optimization, we formulate the problem as follows:

$$\min_{\lambda_1,\lambda_2,\lambda_3} \frac{1}{2}\left|\left|\boldsymbol{y}_V - X_V\hat{\beta} - M_V\hat{\theta}\right|\right|_2^2$$
$$\text{where } \hat{\beta},\hat{\theta} = \arg\min_{\beta,\theta} \frac{1}{2}\left|\left|y - X\beta - M_T\theta\right|\right|_2^2 + \lambda_1||\beta||_1 + \frac{1}{2}\lambda_2||\beta||_2^2 + \frac{1}{2}\lambda_3||D^{(2)}\theta||_2^2 \tag{48}$$

where $M_V \in \mathbb{R}^{|V|\times n}$ is the matrix such that $M_V X = X_V$ and $M_T \in \mathbb{R}^{|T|\times n}$ is the matrix such that $M_T X = X_T$.

The gradients of the validation set error with respect to the regularization parameters are:

$$\nabla_{\lambda_i} L(\boldsymbol{y}_V, X_V\hat{\beta} + \hat{\theta}) = (X_V\frac{\partial\beta}{\partial\lambda_i} + M_V\frac{\partial\theta}{\partial\lambda_i})^T(\boldsymbol{y}_V - X_V\hat{\beta} - \hat{\theta}_V) \tag{49}$$

where

$$\begin{aligned} \frac{\partial\beta}{\partial\lambda_1} &= -(X_T^TX_T - X_T^TM_T(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T + \lambda_2I)^{-1}sign(\hat{\beta}) \\ \frac{\partial\theta}{\partial\lambda_1} &= -(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T\frac{\partial\beta}{\partial\lambda_1} \\ \frac{\partial\beta}{\partial\lambda_2} &= -(X_T^TX_T - X_T^TM_T(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T + \lambda_2I)^{-1}\hat{\beta} \\ \frac{\partial\theta}{\partial\lambda_2} &= -(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T\frac{\partial\beta}{\partial\lambda_2} \\ \frac{\partial\theta}{\partial\lambda_3} &= -(M_T^TM_T - M_T^TX_T(X_T^TX_T + \lambda_2I)^{-1}X_T^TM_T + \lambda_3D^TD)^{-1}D^TD\theta \\ \frac{\partial\beta}{\partial\lambda_3} &= -(X_T^TX_T + \lambda_2I)^{-1}X_T^TM_T\frac{\partial\theta}{\partial\lambda_3} \end{aligned} \tag{50}$$

Acknowledgments

BLAH

References

1. Lorem M, Ipsum VE (1990) Rank Correlation Methods. New York: Oxford University Press, 5th edition.