Regularization Parameter Selection based on Validation-Error Descent with Non-smooth Criterions

Jean Feng*
Department of Biostatistics, University of Washington
and
Noah Simon
Department of Biostatistics, University of Washington

July 21, 2016

Abstract

In high-dimensional and/or non-parametric regression problems, regularization (or penalization) is used to control model complexity and induce desired structure. Each penalty has a weight parameter that indicates how strongly the structure corresponding to that penalty should be enforced. Tuning these parameters is important for model accuracy but is computationally challenge. We propose tuning parameters by solving a continuous optimization problem over a validation set and updating the values using a descent-based approach. We show that the gradient of the validation error with respect to the penalty parameters can be calculated even in regressions with non-smooth penalty functions. The strength of this method is illustrated via simulations and a data analysis.

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

*Jean Feng was supported by NIH grants DP5OD019820 and T32CA206089. Noah Simon was supported by NIH grant DP5OD019820. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

1 Introduction

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

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

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

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

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

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

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

The simplest approach to solving (2) is brute force: one fits models over a grid of parameter values and selects the model with the lowest validation error. As long as the grid is large and fine enough, this method of "grid search" will find a solution close to the global optimum. Unfortunately, it is computationally intractable in cases with more than two parameters since the runtime is exponential in the number of parameters.

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

Various proposals for tuning these "hyperparameters" has been proposed in the machine learning literature. They generally come in two classes: gradient-free and gradient-based methods. The current gold standard is gradient-free Bayesian optimization (Snoek et al. 2012), (Bergstra et al. 2011), (Hutter et al. 2011). However, these methods are effective in situations with no more than twenty or so hyperparameters. Gradient-based methods can tune many more parameters. One approach is to calculate the derivative either by reverse-mode differentiation through the entire training procedure(Maclaurin et al. 2015). Another way is via implicit differentiation of the KKT conditions (Larsen et al. 1998), (Bengio 2000), (Foo et al. 2008), (Lorbert & Ramadge 2010). This latter set of methods assume the function is smooth. Our method shows that the gradient can still be calculated via implicit differentiation in the case of non-smooth criterions.

We compare our approach to grid search and gradient-free methods in simulation studies. Our method has superior performance in situations with many (20+) penalty parameters and is efficient even for regressions with hundreds of penalty parameters. 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.

In Section 2, we describe descent-based optimization and show how to apply it to problems

with non-smooth training criterions. In Section 3, we present simulation studies on problems with relatively few regularization parameters and show that our method achieves validation errors comparable to grid search and gradient-free methods. In Section 4, we solve variants of the example regression problems that have many more regularization parameters with gradient-based descent and show that the models achieve better performance. Finally, we present results on data predicting colitis status from gene expression in Section 5.

Updates to the sections

- Section 2: remove ridge regression cause people might know it already? Add a discussion on the efficiency of the method for nonsmooth criterion. In the case of smooth criterions, the inverse of the Hessian is very expensive to calculate. For us, the inverse of the Hessian is cheap since the Hessian is extremely sparse.
- Section 3: 2 - 5? penalty parameters - should we think about putting APLM with 3 penalties or Nonparametric additive model with 3 covariates?
- Section 4: 30 - 100 penalty parameters - one example or do we add an example on non-parametric additive model with 30 covariates penalized by sparsity (L1 trendfiltering?) or sparsity-smoothness (basis function approach)?
- Section 6: Discussion of overfitting - are we overfitting by adding so many new degrees of freedom? Research TBD.

2 Descent-based Joint Optimization

2.1 Definition

In this manuscript we will restrict ourselves to classes $\mathcal{F} = \{f_{\boldsymbol{\theta}} | \boldsymbol{\theta} \in \Theta\}$, which, for a fixed sample size n, are in some finite dimensional space Θ. This is not a large restriction: the class of linear functions functions meets this requirement; as does any class of finite dimensional parametric functions. Even non-parametric methods generally either use a growing basis expansion (e.g. Polynomial regression, smoothing-splines, wavelet-based-regression, locally-adaptive regression splines (Tsybakov 2008), (Wahba 1981), (Donoho & 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:

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

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

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

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

(4) is the explicit, though often unstated, criterion that training/validation methods attempt to minimize when choosing penalty parameters. The current standard is to minimize this using an exhaustive grid search. Grid-based methods solve the joint optimization problem by fitting models over a J-dimensional grid in the penalty parameter space; so the computational runtime grows exponentially with the number of penalty parameters. While the approach is simple and powerful for a single penalty parameter, optimizing even moderate-dimensional functions (3+) via exhaustive grid search is inefficient (and quickly becomes completely intractable). In addition, (4) is generally a continuous, piecewise-smooth problem. An exhaustive search ignores information available from the smoothness of the surface.

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

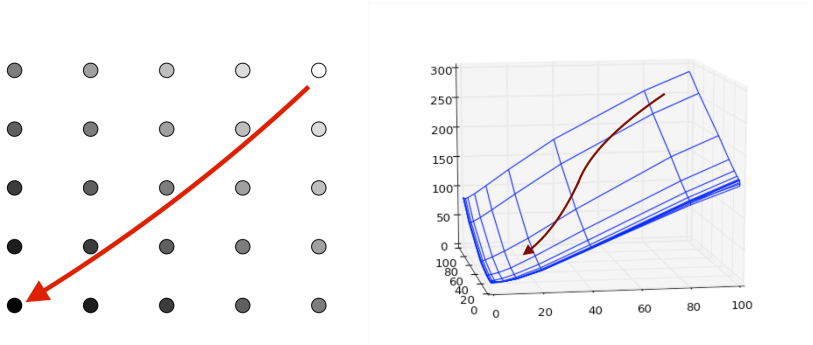

Figure 1: Left: A hypothetical grid of (λ_1, λ_2) points that an exhaustive grid search would fit models for. The darkness of each point indicates the validation cost; dark points mean lower cost. In this example, descent-based optimization would takes steps along the arrow, while a grid search would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

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

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

2.2 Smooth Training Criterion

Let us denote the training criterion as follows

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

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

Algorithm 1 Gradient Descent for Smooth Training Criterions

Initialize $\boldsymbol{\lambda}^{(0)}$.

for each iteration $k = 0, 1, ...$ until stopping criteria is reached **do**

Perform gradient step with step size $t^{(k)}$

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

end for

There are a number of potential ways to choose the step-size $t^{(k)}$. Choice of step-size is discussed further in Section 2.5.

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

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

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

$$\nabla_\theta L_T(\boldsymbol{\theta}, \boldsymbol{\lambda})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})} = \mathbf{0}. \tag{8}$$

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

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = -\left[\nabla_\theta^2 L_T(\boldsymbol{\theta}, \boldsymbol{\lambda})^{-1} \nabla_{\boldsymbol{\theta}} P(\boldsymbol{\theta})\right]\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})} \tag{9}$$

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

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

2.3 Nonsmooth Training Criterion

When the penalized training criterion in the joint optimization problem is not smooth, gradient descent cannot be applied. Nonetheless, we find that in many problems, the solution

$\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is smooth at almost every $\boldsymbol{\lambda}$ (eg. Lasso, Group Lasso, Trend Filtering); this means that we can indeed apply gradient descent in practice. In this section, we characterize these problems that are almost everywhere smooth. In addition, we provide a solution for deriving $\frac{\partial}{\partial \boldsymbol{\lambda}}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ since calculating the gradient is a challenge in and of itself. This is then incorporated into an algorithm for tuning $\boldsymbol{\lambda}$ using gradient descent.

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

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

$$\Omega^{L}(\boldsymbol{\eta}) = \left\{ \boldsymbol{u} \middle| \lim_{\epsilon \to 0} \frac{L(\boldsymbol{\eta} + \epsilon \boldsymbol{u}) - L(\boldsymbol{\eta})}{\epsilon} \ \textit{exists} \right\} \tag{10}$$

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

$$\arg\min_{\boldsymbol{\theta} \in \Theta} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta} \in S} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{11}$$

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

$$_{\boldsymbol{B}}\nabla f = \begin{pmatrix} \frac{\partial f}{\partial \boldsymbol{b}_1} \\ \frac{\partial f}{\partial \boldsymbol{b}_2} \\ \vdots \\ \frac{\partial f}{\partial \boldsymbol{b}_p} \end{pmatrix} \in \mathbb{R}^p; \quad _{\boldsymbol{B}}\nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial b_1^2} & \frac{\partial^2 f}{\partial b_1 \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_1 \partial b_p} \\ \frac{\partial^2 f}{\partial b_2 \partial b_1} & \frac{\partial^2 f}{\partial b_2^2} & \cdots & \frac{\partial^2 f}{\partial b_2 \partial b_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial b_p \partial b_1} & \frac{\partial^2 f}{\partial b_p \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_p^2} \end{pmatrix} \in \mathbb{R}^{p \times p} \tag{12}$$

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

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

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

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

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

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

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

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

where

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

We can therefore construct a gradient descent procedure based on the model parameter constraint in (14). At each iteration, let matrix $\boldsymbol{U}$ have orthonormal columns spanning the differentiable space $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$. Since this space is also a local optimality space, it is sufficient to minimize the training criterion over the column space of $\boldsymbol{U}$. The joint optimization problem can be reformulated using $\boldsymbol{U}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ as the model parameters instead:

$$\begin{aligned} &\min_{\boldsymbol{\lambda}\in\Lambda} L(\boldsymbol{y}_V, f_{\boldsymbol{U}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ &\text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} L_T(\boldsymbol{U}\boldsymbol{\beta},\boldsymbol{\lambda}) \end{aligned} \tag{15}$$

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

$$\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \boldsymbol{U}\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \tag{16}$$

Note that because the differentiable space is a local optimality space and is thus locally constant, we can treat $\boldsymbol{U}$ as a constant in the gradient derivations. Algorithm 2 provides the exact steps for tuning the regularization parameters.

Algorithm 2 Joint Optimization with Gradient Descent

Initialize $\boldsymbol{\lambda}^{(0)}$.

for each iteration $k = 0, 1, ...$ until stopping criteria is reached **do**

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

Construct matrix $\boldsymbol{U}^{(k)}$, an orthonormal basis of $\Omega^{L_T(\cdot, \boldsymbol{\lambda})}\left(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k)})\right)$.

Define the locally equivalent joint optimization problem

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \Lambda} L(\boldsymbol{y}_V, f_{\boldsymbol{U}^{(k)}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ &\text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} L_T(\boldsymbol{U}^{(k)}\boldsymbol{\beta}, \boldsymbol{\lambda}) \end{aligned}$$

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

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\left[\left({}_{\boldsymbol{U}^{(k)}}\nabla^2 L_T(\boldsymbol{U}^{(k)}\boldsymbol{\beta}, \boldsymbol{\lambda})\right)^{-1} {}_{\boldsymbol{U}^{(k)}}\nabla P(\boldsymbol{U}^{(k)}\boldsymbol{\beta})\right]\Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}$$

with ${}_{\boldsymbol{U}^{(k)}}\nabla^2$ and ${}_{\boldsymbol{U}^{(k)}}\nabla$ are as defined in (12).

Calculate the gradient $\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\theta}(\boldsymbol{\lambda})}(\boldsymbol{X_V}))|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}}$ where

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

Perform the gradient update with step size $t^{(k)}$

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

end for

Thus far we have restricted our attention to joint optimization for training/validation splits. Joint optimization for K-fold cross validation is described in the Appendix.

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 examples where the training criterions are nonsmooth, but $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is smooth almost everywhere.

For ease of notation, we will let $S_{\boldsymbol{\lambda}}$ denote the differentiable space of $L_T(\cdot, \boldsymbol{\lambda})$ at $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$. All the example regressions satisfy the conditions in Theorem 1; details are included in the Appendix. Note that in some 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 is

$$\begin{aligned} &\min_{\lambda \in \mathbb{R}_+} \tfrac{1}{2}\|\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\lambda)\|_2^2 \\ &\text{s.t. } \hat{\boldsymbol{\theta}}(\lambda) = \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}\|_2^2 + \tfrac{1}{2}\lambda \|\boldsymbol{\theta}\|_2^2 \end{aligned} \tag{17}$$

The KKT conditions state that $\hat{\boldsymbol{\theta}}(\lambda)$ must satisfy

$$-\boldsymbol{X}_T^\top (\boldsymbol{y}_T - \boldsymbol{X}_T \hat{\boldsymbol{\theta}}(\lambda)) + \lambda \hat{\boldsymbol{\theta}}(\lambda) = 0 \tag{18}$$

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.

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

2.4.2 Elastic Net

The elastic net (Zou & 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:

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \tfrac{1}{2}\|\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\|^2 \\ &\text{s.t. } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}\|^2 + \lambda_1 \|\boldsymbol{\theta}\|_1 + \tfrac{1}{2}\lambda_2 \|\boldsymbol{\theta}\|_2^2 \end{aligned} \tag{20}$$

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

Let $\boldsymbol{X}_{T,I(\boldsymbol{\lambda})} = \boldsymbol{X}_T \boldsymbol{I}_{I(\boldsymbol{\lambda})}$ and $\boldsymbol{X}_{V,I(\boldsymbol{\lambda})} = \boldsymbol{X}_V \boldsymbol{I}_{I(\boldsymbol{\lambda})}$. The locally equivalent joint optimization problem is

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \mathbb{R}^2_+} \tfrac{1}{2} \|\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\|^2 \\ \text{s.t. } &\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \tfrac{1}{2} \|\boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} \boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \tfrac{1}{2} \lambda_2 \|\boldsymbol{\beta}\|_2^2 \end{aligned} \tag{21}$$

To calculate the gradient, we can apply (9) since the training criterion is now smooth

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\left(\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}^\top \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} + \lambda_2 \boldsymbol{I}\right)^{-1} \left[sgn\left(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right) \quad \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right] \tag{22}$$

Hence, the gradient of the validation loss with respect to $\boldsymbol{\lambda}$ is

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\lambda)}(\boldsymbol{X_V})) = -\left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right) \tag{23}$$

2.4.3 Sparse Group Lasso

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

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

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \mathbb{R}^2_+} \tfrac{1}{2n} \left\| \boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \\ \text{s.t. } &\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2n} \|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}\|_2^2 + \lambda_1 \sum_{m=1}^M \|\boldsymbol{\theta}^{(m)}\|_2 + \lambda_2 \|\boldsymbol{\theta}\|_1 + \tfrac{1}{2} \epsilon \|\boldsymbol{\theta}\|_2^2 \end{aligned} \tag{24}$$

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

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

$$\begin{aligned} &\min_{\boldsymbol{\lambda}\in\mathbb{R}^2_+} \tfrac{1}{2n}\left\|\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right\|_2^2 \\ \text{s.t. } &\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \tfrac{1}{2n}\left\|\boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})}\boldsymbol{\beta}\right\|_2^2 + \lambda_1 \textstyle\sum_{m=1}^{M} \|\boldsymbol{\beta}^{(m)}\|_2 + \lambda_2\|\boldsymbol{\beta}\|_1 + \tfrac{1}{2}\epsilon\|\boldsymbol{\beta}\|_2^2 \end{aligned} \tag{25}$$

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

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X_V})) = -\frac{1}{n}\left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right)^\top\left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right) \tag{26}$$

where

$$\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\left(\frac{1}{n}\boldsymbol{X}^\top_{T,I(\boldsymbol{\lambda})}\boldsymbol{X}_{T,I(\boldsymbol{\lambda})} + \lambda_1\boldsymbol{B}(\boldsymbol{\lambda}) + \epsilon\boldsymbol{I}_p\right)^{-1}\begin{bmatrix}\begin{bmatrix}\frac{\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})||_2}\\ \dots \\ \frac{\hat{\boldsymbol{\beta}}^{(M)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(M)}(\boldsymbol{\lambda})||_2}\end{bmatrix} & sgn(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}))\end{bmatrix} \tag{27}$$

2.4.4 Additive Partially Linear Models

Finally consider an additive partially linear model (APLM) for response y given covariates $\boldsymbol{x} \in \mathbb{R}^p$ and $\boldsymbol{z} \in \mathbb{R}^q$. For this semi-parametric regression we assume that y is the sum of p univariate functions and a linear function:

$$y = \sum_{i=1}^{p} f_i(x_i) + \boldsymbol{\beta}^\top\boldsymbol{z} + \epsilon \tag{28}$$

We want to estimate the coefficients $\boldsymbol{\beta}$ and values of the functions f_i at our observations: $\boldsymbol{\theta}^{(i)} \equiv (f_i(x_{i1}), ..., f_i(x_{in}))$. As before the model is fit using least squares. In addition, a smoothness penalty is applied to $\hat{\boldsymbol{\theta}}^{(i)}$ and a lasso penalty is applied to $\hat{\boldsymbol{\beta}}$ in the training criterion.

Let $\boldsymbol{Z}_T \in \mathbb{R}^{|T|\times q}, \boldsymbol{Z}_V \in \mathbb{R}^{|V|\times q}$ be the linear covariates from the training and validation sets and $\boldsymbol{X} \in \mathbb{R}^{n\times q}$ be the nonlinear covariates. Define matrices $\boldsymbol{I}_T$ and $\boldsymbol{I}_V$ such that $\boldsymbol{I}_T\boldsymbol{\theta}^{(i)}$ and $\boldsymbol{I}_V\boldsymbol{\theta}^{(i)}$ are estimates for f_i at the training and validation inputs, respectively. The joint optimization problem is

$$\begin{aligned} &\min_{\boldsymbol{\lambda}\in\mathbb{R}^{p+1}_+} \tfrac{1}{2}\left\|\boldsymbol{y}_V - \boldsymbol{Z}_V\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \boldsymbol{I}_V\textstyle\sum_{i=1}^{p}\hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda})\right\|_2^2 \\ \text{s.t. } &\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta},\boldsymbol{\theta}} \tfrac{1}{2}\left\|\boldsymbol{y}_T - \boldsymbol{Z}_T\boldsymbol{\beta} - \boldsymbol{I}_T\textstyle\sum_{i=1}^{p}\boldsymbol{\theta}^{(i)}\right\|_2^2 + \lambda_0\|\boldsymbol{\beta}\|_1 \\ &\quad + \tfrac{1}{2}\textstyle\sum_{i=1}^{p}\lambda_i\left\|\boldsymbol{D}^{(2)}_{\boldsymbol{x}_i}\boldsymbol{\theta}^{(i)}\right\|_2^2 + \tfrac{1}{2}\epsilon\left(\|\boldsymbol{\beta}\|_2^2 + \textstyle\sum_{i=1}^{p}\|\boldsymbol{\theta}^{(i)}\|_2^2\right) \end{aligned} \tag{29}$$

The matrix $\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}$ gives the second-order differences between the estimates of f_i for unevenly-spaced inputs. Construction of $\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}$ is given in the Appendix. Note that the nonlinear covariates $\boldsymbol{X}$ from the training and validation sets are incorporated into the matrices $\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}$ in order to estimate f_i at all $\boldsymbol{X}$ values.

The formulation in (29) is given in a very general form. In the following sections we will consider special cases of this regression, such as pooling certain λ_i's and removing the parametric component altogether.

The gradient derivation is given in the Appendix. The logic is the same as the other examples.

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 & 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 $\alpha \in [0.001, 0.01]$ and $\beta \in [0.01, 0.1]$. Details of backtracking line search are given in the Appendix. During gradient descent, it is possible that the step size will result in a negative regularization parameter; we reject any step that would set a regularization parameter to below a minimum threshold of $1e$-8.

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

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

for some prespecified tolerance δ. For the results in this manuscript we use $\delta \in [1e\text{-}4, 1e\text{-}5]$.

3 Results: regressions with two penalty parameters

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

The regularization parameters were tuned over a training/validation split. Descent-based joint optimization was implemented using gradient descent. The inner training criterions were solved using the splitting conic solver (SCS) or ECOS in CVXPY (Diamond & Boyd 2016), depending on the accuracy of the solver for the particular problem. The simulation settings are described below, followed by a discussion of the results.

3.1 Simulation settings

3.1.1 Elastic Net

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

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \sigma\boldsymbol{\epsilon} \text{ where } \boldsymbol{\beta} = (\underbrace{1, ..., 1}_{\text{size 15}}, \underbrace{0, ..., 0}_{\text{size 235}}) \tag{30}$$

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

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

3.1.2 Sparse group lasso

We generated thirty datasets, each consisting of 60 training and 20 validation observations with 300 covariates. The predictors $\boldsymbol{X}$ were generated from a standard normal distribution. The response $\boldsymbol{y}$ was generated by

$$\boldsymbol{y} = \sum_{j=1}^{3} \boldsymbol{X}^{(j)}\boldsymbol{\beta}^{(j)} + \sigma\boldsymbol{\epsilon} \text{ where } \boldsymbol{\beta}^{(j)} = (1, 2, 3, 4, 5, 0, ..., 0) \tag{31}$$

where $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I})$. σ was chosen such that the signal to noise ratio was 2. The group lasso penalty was specified using $M = 30$ covariate groups with 10 covariates each.

Both descent-based methods were initialized at (0.1, 0.1) and (1, 1). Grid search was performed over a 10×10 grid from $1e$-5 to $\max(\{||\boldsymbol{X}^{(j)T}\boldsymbol{y}||_2\}_{j=1,2,3})$.

3.1.3 Additive partially linear models

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

$$\begin{aligned} x_{ij} &= Z_1 + \delta_{ij} \text{ for } j = 1, 2, 3 \\ x_{ij} &= Z_2 + \delta_{ij} \text{ for } j = 4, 5, 6 \\ x_{ij} &\sim N(0, 1) \text{ for } j = 7, ..., 20 \end{aligned} \tag{32}$$

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

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \kappa g(\boldsymbol{z}) + \sigma\boldsymbol{\epsilon} \tag{33}$$

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

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

3.2 Discussion of results

As shown in Table 1, the descent-based joint optimization and grid search do not have significantly different performance in terms of minimizing the validation error. Descent-based joint optimization is also faster than grid search in all three examples.

4 Results: regressions with more than two penalty parameters

In this second simulation study we tested descent-based joint optimization on regressions with more than two regularization parameters. We compared the performance of regressions models fit with $k > 2$ penalty terms to that those fit with $k \leq 2$ penalty terms.

Table 1: Validation errors for simulation studies in Section 3. Standard errors provided in parentheses.

Elastic Net		
	Validation Error	Runtime (sec)
Grid Search	114.61 (9.04)	67.60
Gradient Descent	113.95 (9.43)	14.77
Sparse Group Lasso		
	Validation Error	Runtime (sec)
Grid Search	47.81 (3.78)	78.26
Gradient Descent	46.66 (3.79)	7.45
APLM		
	Validation Error	Runtime (sec)
Grid Search	3.39 (0.17)	55.27
Gradient Descent	3.45 (0.17)	16.04

We experimented with generalizations of the simple regressions from the previous section. For the sparse group lasso, we tried an "un-pooled" version in which each covariate group has its own regularization parameter. For the additive partial linear model example, we added a ridge penalty as a third regularization term. In addition, we consider a special case of the APLM that is completely nonparametric. Joint optimization formulations and gradient derivations for these generalized regression models are in the Appendix.

Our results show that the models from the generalized regressions achieved lower test error and tuning their regularization parameters using gradient descent was computationally tractable, even in cases with a hundred regularization parameters.

4.1 Un-pooled sparse group lasso

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

group lasso defines the training criterion as follows:

$$L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}) \equiv \frac{1}{2n} \|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\|_2^2 + \sum_{m=1}^{M} \lambda_1^{(m)} \|\boldsymbol{\theta}^{(m)}\|_2 + \lambda_2 \|\boldsymbol{\theta}\|_1 + \frac{1}{2}\epsilon \|\boldsymbol{\theta}\|_2^2 \tag{34}$$

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

We ran three experiments with different numbers of covariate groups M and total covariates p, as given in Table 2. The simulation settings were similar to the simulation settings and grid search procedure in Section 3.1.2. The validation set was one third the size of the training set and a separate test set of 200 observations was generated each run. For gradient descent, the $M + 1$ regularization parameters were initialized at $0.1 \times \mathbf{1}$ and $\mathbf{1}$.

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

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

4.2 Additive partially linear model with three penalties

As in the previous APLM simulation study in Section 3.1.3, we suppose there is one nonlinear covariate and 20 linear predictors. We now fit an APLM using an the elastic net penalty instead of the lasso for $\boldsymbol{\beta}$. For this example, the training criterion is then

$$L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}) \equiv \frac{1}{2} \|\boldsymbol{y}_T - \boldsymbol{Z}_T\boldsymbol{\beta} - \boldsymbol{I}_T\boldsymbol{\theta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \frac{1}{2}\lambda_2 \|\boldsymbol{\beta}\|_2^2 + \frac{1}{2}\lambda_3 \|\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}\boldsymbol{\theta}\|_2^2 + \frac{1}{2}\epsilon \|\boldsymbol{\theta}\|_2^2 \tag{35}$$

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

Table 2: Un-pooled sparse group lasso and sparse group lasso (SGL) tuned by gradient descent and grid search, respectively. Standard errors given in parentheses.

n=60, p=300, g=3, M=30				
	β Error	% Correct Nonzero β	Test Error	Runtime (sec)
SGL	8.24 (0.26)	12.48 (1.85)	57.34 (2.43)	78.14
Un-pooled SGL	7.36 (0.37)	21.94 (2.21)	50.93 (3.10)	16.63
n=90, p=900, g=3, M=60				
	β Error	% Correct Nonzero β	Test Error	Runtime (sec)
SGL	8.26 (0.24)	9.41 (0.56)	54.43 (2.42)	324.54
Un-pooled SGL	6.90 (0.24)	11.48 (0.72)	43.92 (2.00)	351.68
n=90, p=1200, g=3, M=100				
	β Error	% Correct Nonzero β	Test Error	Runtime (sec)
SGL	8.30 (0.23)	9.81 (0.81)	57.07 (2.10)	448.92
Un-pooled SGL	6.67 (0.33)	13.19 (0.97)	46.09 (2.65)	337.70

We used the same simulation settings as those in Section 3.1.3. A separate test set of 200 observations was generated each run. Gradient descent was initialized at $\boldsymbol{\lambda} = 10^i \times \mathbf{1}$ for $i = -2, ..., 1$. We experimented with three nonlinear functions $g : \mathbb{R} \mapsto \mathbb{R}$ of varying levels of smoothness, as given in Table 3.

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

The results show that the generalized APLM criterion had better performance by all three metrics in most cases. The linear model fits improved the most, which supports our hypothesis.

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

Table 3: APLM with three penalties (APLM 3) vs. two penalties (APLM 2). Penalty parameters were tuned by gradient descent and grid search, respectively. Standard errors given in parentheses.

$g(z) = 4z^3 - z^2 + 2z$				
	β Error	θ Error	Test Error	Runtime (sec)
APLM 2	0.52 (0.02)	3.41 (0.17)	29.39 (0.76)	67.87
APLM 3	0.31 (0.06)	3.29 (0.14)	28.93 (0.68)	80.67
$g(z) = \sin(5z) + \sin(15(z-3))$				
	β Error	θ Error	Test Error	Runtime (sec)
APLM 2	0.59 (0.05)	3.95 (0.24)	31.16 (1.45)	63.48
APLM 3	0.30 (0.05)	3.67 (0.15)	29.80 (0.69)	72.50
$g(z) = (2-z)\sin(20z^4)$				
	β Error	θ Error	Test Error	Runtime (sec)
APLM 2	0.49 (0.02)	4.43 (0.12)	32.22 (0.82)	67.75
APLM 3	0.32 (0.03)	4.45 (0.12)	31.77 (0.71)	80.81

Table 4: Additive model with three penalty parameters (AM 3) vs. one penalty parameter (AM 1). The parameters were tuned by gradient descent and grid search, respectively. Standard errors given in parentheses.

	Validation Error	Test Error	Runtime (sec)
AM 3	33.97 (1.31)	38.17 (1.61)	97.15
AM 1	38.93 (1.57)	42.84 (1.88)	23.70

4.3 Nonparametric Additive Models

Finally, we consider the special case of APLMs where a completely nonparametric model is used for y. That is, suppose that y is the sum of p univariate functions f_i, where each is penalized separately with parameter λ_i. This model is particularly useful when the functions have varying levels of smoothness. Ideally, λ_i is large for f_i with nearly constant first-order derivatives and small for f_i with drastically changing first-order derivatives.

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

$$y = \sum_{i=1}^{3} f_i(x_i) + \sigma\epsilon \tag{36}$$

where $\epsilon \sim N(0, 1)$ and σ was chosen such that the signal to noise ratio was 2. The univariate functions in (36) are

$$\begin{aligned} f_1(x_1) &= 9\sin(2x_1) \\ f_2(x_2) &= x_2 \\ f_3(x_3) &= 6\cos(1.25x_3) + 6\sin(0.5x_3 + 0.5) \end{aligned} \tag{37}$$

For baseline comparison, we fit the model with a single λ and tuned it using grid search over 10 log-spaced values from $1e$-3 to 50. Gradient descent was initialized at $\lambda_1 = \lambda_2 = \lambda_3 = 1$.

As seen in Table 4, the additive model with three penalty parameters had significantly lower validation error compared to that with a single penalty parameter. Furthermore, the additive model with three penalty parameters achieved significantly lower error on a separate test set. It is interesting to note that gradient descent consistently determined that f_2 was the smoothest function; for 27 of the thirty runs, $\hat{\lambda}_2$ was larger than $\hat{\lambda}_1$ and $\hat{\lambda}_3$.

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

5 Application to Biological Data

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

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

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

Table 5 presents the average results from repeating this process ten times. Un-pooled sparse group lasso achieved a slightly higher classification rate than sparse group lasso. Interestingly, un-pooled sparse group lasso found solutions that were significantly more sparse than sparse group lasso; on average, un-pooled sparse group lasso identified 9 genesets whereas sparse group lasso identified 38. These results suggest that un-pooling the penalty parameters in sparse group lasso could potentially improve interpretability.

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

6 Discussion

In this paper, we proposed finding the optimal regularization parameters by treating it as an optimization problem over the regularization parameter space. We have proven that

Table 5: Predictive genes and genesets of Ulcerative Colitis found by un-pooled sparse group lasso vs. sparse group lasso (SGL). Standard errors given in parenthesis.

	% Correct	Num. Genesets	Num. Genes	Runtime (sec)
SGL	82.47 (0.26)	38.4 (8.19)	207.0 (47.12)	2722.4
Un-pooled SGL	84.29 (0.17)	8.9 (0.44)	83.9 (8.15)	2298.5

a descent-based approach can be used for regression problems in which the penalties are smooth almost everywhere and present a general algorithm for performing a modified gradient descent.

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

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

References

Bengio, Y. (2000), 'Gradient-based optimization of hyperparameters', *Neural computation* **12**(8), 1889–1900.

Bergstra, J. S., Bardenet, R., Bengio, Y. & Kégl, B. (2011), Algorithms for hyper-parameter optimization, *in* 'Advances in Neural Information Processing Systems', pp. 2546–2554.

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

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

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

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

Foo, C.-s., Do, C. B. & Ng, A. Y. (2008), Efficient multiple hyperparameter learning for log-linear models, *in* 'Advances in neural information processing systems', pp. 377–384.

Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2011), Sequential model-based optimization for general algorithm configuration, *in* 'International Conference on Learning and Intelligent Optimization', Springer, pp. 507–523.

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

Larsen, J., Svarer, C., Andersen, L. N. & Hansen, L. K. (1998), Adaptive regularization in neural network modeling, *in* 'Neural Networks: Tricks of the Trade', Springer, pp. 113–132.

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

Maclaurin, D., Duvenaud, D. & Adams, R. P. (2015), Gradient-based hyperparameter optimization through reversible learning, *in* 'Proceedings of the 32nd International Conference on Machine Learning'.

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

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

Snoek, J., Larochelle, H. & Adams, R. P. (2012), Practical bayesian optimization of machine learning algorithms, *in* 'Advances in neural information processing systems', pp. 2951–2959.

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

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

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

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

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