

---

# 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:

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

where  $L$  is the loss function,  $\{P_i\}_{i=1, \dots, J}$  are the penalty functions, and  $\{\lambda_i\}_{i=1, \dots, J}$  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  $(X_T, y_T)$  and measures the model's error on a validation set  $(X_V, y_V)$ . The goal then is to choose penalty parameters that minimize the validation error, as formulated in the following optimization problem:

$$\begin{aligned} & \min_{\lambda \in \Lambda} L(y_V, \hat{f}(X_V | \lambda)) \\ & \text{where } \hat{f}(\cdot | \lambda) = \arg \min_f L(y_T, f(X_T)) + \sum_{i=1}^J \lambda_i P_i(f) \end{aligned} \quad (2)$$

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  $\mathbf{y}$  be the  $n$  response vector and  $\mathbf{X}$  be the  $n \times p$  design matrix.

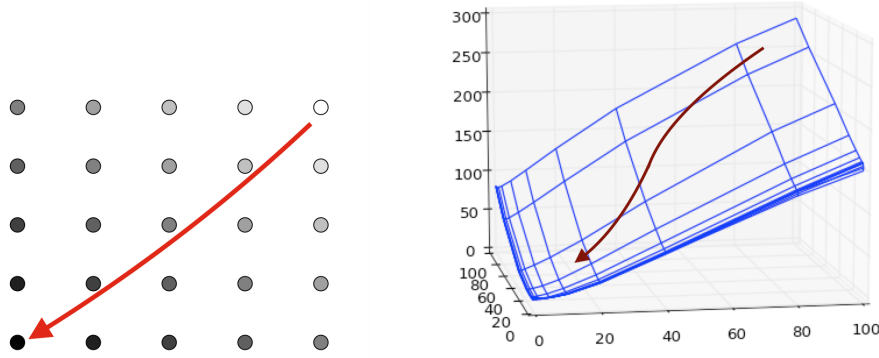
We are interested in finding the optimal model  $f$  in some model class  $F$  that minimizes a penalized criterion of the form:

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

where  $L : \mathbb{R}^{2n} \rightarrow \mathbb{R}$  is the loss function,  $P_i : \mathbb{R}^{m_i} \rightarrow \mathbb{R}$  is the  $i$ th penalty function, and  $\lambda_i$  are the corresponding penalty parameters. Since the optimization problem in (3) only requires fitted values at the points in the design matrix  $\mathbf{X}$  and the penalty functions take in finite-dimensional inputs, the optimization problem reduces to optimizing over some finite-dimensional space  $\Theta$ . If  $F$  is the space of parametric models, then  $\Theta$  would be the space of model parameters. If  $F$  is the space of nonparametric models, then  $\Theta$  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:

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

Suppose that we use a training/validation split to select penalty parameters  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_k)^T$ . Let the data be partitioned into a training set  $(\mathbf{y}_T, \mathbf{X}_T)$  and validation set  $(\mathbf{y}_V, \mathbf{X}_V)$ . 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:

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

(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  $k$ -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.

## 2.2 Algorithm

Many joint optimization problems can be solved using gradient descent. The procedure is relatively simple, at least in principle. To find the regularization parameters  $\lambda$  that minimize the validation

criterion  $L(\mathbf{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\mathbf{X}_V))$ , we initialize gradient descent at some point  $\boldsymbol{\lambda}^{(0)}$  and take a step in the direction opposite of the gradient at each iteration.

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

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

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

The primary difficulty is determining if the gradient exists and how to calculate it. In this section, we present a theorem that states the conditions under which gradient descent can solve a joint optimization problem and a procedure for deriving the gradient.

Gradient descent can be used whenever the criterion is differentiable almost everywhere. For this paper, we are only interested in finding when  $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda})$  exists since the loss function  $L(y, f_{\theta}(X))$  is usually differentiable with respect to  $\theta$ . If  $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\theta}(\boldsymbol{\lambda})$  exists, the gradient descent direction can be calculated using the chain rule:

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

Recall that  $\hat{\theta}(\boldsymbol{\lambda})$  is the minimizer of the training criterion:

$$\hat{\theta}(\boldsymbol{\lambda}) = \arg \min_{\theta \in \Theta} L(\mathbf{y}_T, f_{\theta}(\mathbf{X}_T)) + \sum_{i=1}^J \lambda_i P_i(\theta) \quad (8)$$

Our approach is to find a locally equivalent function  $\tilde{\theta}(\boldsymbol{\lambda})$  that is the minimizer of the training criterion over a subspace  $\tilde{\Theta}(\boldsymbol{\lambda})$  in which the training criterion is twice continuously differentiable. That is, for a given  $\boldsymbol{\lambda}$ , we find some neighborhood  $W$  containing  $\boldsymbol{\lambda}$  such that for every  $\boldsymbol{\lambda}' \in W$

$$\tilde{\theta}(\boldsymbol{\lambda}') = \arg \min_{\theta \in \tilde{\Theta}(\boldsymbol{\lambda})} L(\mathbf{y}_T, f_{\theta}(\mathbf{X}_T)) + \sum_{i=1}^J \lambda'_i P_i(\theta) = \hat{\theta}(\boldsymbol{\lambda}') \quad (9)$$

Finding the gradient descent direction is straightforward once we have the twice differentiable function  $\tilde{\theta}(\boldsymbol{\lambda})$ . To find  $\frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\theta}(\boldsymbol{\lambda})$ , we perform implicit differentiation on the gradient condition

$$\nabla_{\theta} \left( L(\mathbf{y}_T, f_{\theta}(\mathbf{X}_T)) + \sum_{i=1}^J \lambda_i P_i(\theta) \right) \Big|_{\theta=\tilde{\theta}(\boldsymbol{\lambda})} = \vec{0} \quad (10)$$

and, assuming the hessian of the training criterion is nonsingular, we get the following equation

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\theta}(\boldsymbol{\lambda}) = - \left[ \nabla_{\theta}^2 \left( L(\mathbf{y}_T, f_{\theta}(\mathbf{X}_T)) + \sum_{i=1}^J \lambda_i P_i(\theta) \right) \right]^{-1} \begin{pmatrix} \nabla_{\theta} P_1(\theta) & \nabla_{\theta} P_2(\theta) & \dots & \nabla_{\theta} P_J(\theta) \\ \vdots & \vdots & & \vdots \end{pmatrix} \Big|_{\theta=\tilde{\theta}(\boldsymbol{\lambda})} \quad (11)$$

Plugging (11) into (7) gives us the gradient descent direction.

We now present the formal requirements for the gradient to exist almost everywhere. First, we define some terminology.

**Definition 1.** Given a real-valued function  $L$  and a point  $\eta$  in its domain, let  $U$  be the set of vectors such that the directional derivatives of  $L$  at the point  $\eta$  along  $u \in U$  exist. The space spanned by  $U$  is the “differentiable space” of  $L$  at  $\eta$ :

$$\Theta^{(L)}(\eta) = \text{span} \left( \left\{ \delta \mid \lim_{\epsilon \rightarrow 0} \frac{L(\eta + \epsilon \delta) - L(\eta)}{\epsilon} \text{ exists} \right\} \right) \quad (12)$$

**Definition 2.**  $S$  is a “local optimality space” for a real-valued function  $L(\theta, \lambda)$  within neighborhood  $W$  if for every  $\lambda \in W$ , the minimizer of  $L(\cdot, \lambda)$  is in the space  $S$

$$\arg \min_{\theta} L(\theta, \lambda) = \arg \min_{\theta \in S} L(\theta, \lambda) \quad (13)$$

The gradient exists almost everywhere if three conditions are satisfied. For notational convenience, let the training criterion be denoted as

$$L_T(\theta, \lambda) = L(\mathbf{y}_T, f_{\theta}(\mathbf{X}_T)) + \sum_{i=1}^J \lambda_i P_i(\theta) \quad (14)$$

First, we require that for almost every  $\lambda$ , the differentiable space of  $L_T(\cdot, \lambda)$  at  $\hat{\theta}(\lambda)$  is a local optimality space for some neighborhood  $W$  containing  $\lambda$ , or stated mathematically as:

$$\arg \min_{\theta} L_T(\theta, \lambda') = \arg \min_{\theta \in \Omega^{L_T(\cdot, \lambda)}(\hat{\theta}(\lambda))} L_T(\theta, \lambda') \forall \lambda' \in W \quad (15)$$

This condition formalizes the previously mentioned  $\tilde{\Theta}(\lambda)$  and ensures that we can always restrict gradient descent to a smooth space.

For example, the lasso regression satisfies this condition. The lasso penalty is defined as

$$P(\theta) = \sum_{i=1}^p |\theta_i| \quad (16)$$

Then  $L_T(\cdot, \lambda)$  at  $\hat{\theta}(\lambda)$  is differentiable only along the directions spanned by the nonzero elements of  $\hat{\theta}(\lambda)$ . Furthermore, lasso regression sets elements of  $\hat{\theta}(\lambda)$  to zero as  $\lambda$  increases and it has been shown that there exists a finite number of  $\lambda$  values in which this transition occurs. In other words, for almost every  $\lambda$ , there is some neighborhood  $W$  in which the nonzero indices of  $\hat{\theta}(\lambda)$  stay constant. Therefore, the differentiable space is a local optimality space. Other joint optimization problems that satisfy this condition include the generalized lasso and all the examples used in this paper.

Finally, we have two other conditions to ensure that (11) is valid. For almost every  $\lambda$ , the training criterion  $L_T$  restricted to  $\Omega^{L_T}(\hat{\theta}(\lambda), \lambda)$  must be twice continuously differentiable and  $\nabla_{\theta}^2 L_T(\theta, \lambda)$  must be invertible.

Under these three conditions, we can prove that the validation loss  $L(\mathbf{y}_V, f_{\hat{\theta}(\lambda)}(\mathbf{X}_V))$  is continuously differentiable almost everywhere. The proof for the following theorem is included in Appendix A.

**Theorem 1.** *Let the optimization problem have the form in (5) and let  $L_T(\theta, \lambda)$  denote the training set criterion.*

*Suppose that the validation loss  $L(\mathbf{y}_V, f_{\hat{\theta}(\lambda)}(\mathbf{X}_V))$  is continuously differentiable with respect to  $\hat{\theta}(\lambda)$ .*

*For almost every  $\lambda_0$ , suppose there exists some neighborhood  $W$  containing  $(\hat{\theta}(\lambda_0), \lambda_0)$  such that:*

- *Within neighborhood  $W$ ,  $L_T(\theta, \lambda)$  is twice continuously differentiable along directions in the differentiable space  $\Omega^{(L_T)}(\hat{\theta}(\lambda_0), \lambda_0)$ .*
- *The differentiable space  $\Omega^{(L_T)}(\hat{\theta}(\lambda_0), \lambda_0)$  is a local optimality space of  $L_T(\cdot, \lambda)$  for neighborhood  $W$ .*

*In addition, suppose that the hessian of  $L_T(\cdot, \lambda_0)$  with respect to the basis of  $\Omega^{(L_T)}(\hat{\theta}(\lambda_0), \lambda_0)$  is nonsingular.*

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

$$\nabla_{\lambda} L(\mathbf{y}_V, f_{\hat{\theta}(\lambda)}(\mathbf{X}_V)) = \left[ \frac{\partial}{\partial \theta} L(\mathbf{y}_V, f_{\hat{\theta}}(\mathbf{X}_V)) \Big|_{\theta=\tilde{\theta}(\lambda)} \right]^T \frac{\partial}{\partial \lambda} \tilde{\theta}(\lambda) \quad (17)$$

where

$$\tilde{\theta}(\lambda) = \arg \min_{\theta \in \Omega^{(L_T)}(\hat{\theta}(\lambda), \lambda)} L_T(\theta, \lambda) \quad (18)$$

As long as a joint optimization problem satisfies conditions described in the theorem, we can use gradient descent to solve it. As before, gradient descent is initialized at some point  $\boldsymbol{\lambda}^{(0)}$ . To calculate the gradient at each iteration  $k$ , we solve the inner optimization problem for  $\hat{\theta}(\boldsymbol{\lambda}^{(k)})$ , which determines the differentiable space of the training criterion at that point. We then consider the restricted minimization problem  $\tilde{\theta}(\boldsymbol{\lambda}^{(k)})$  which has a differentiable criterion by construction. Using (11), we calculate  $\frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\theta}(\boldsymbol{\lambda})|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}}$  and obtain the gradient descent direction using the chain rule. The gradient descent method is described in 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  $\Omega^{(k)}$  be the differentiable space of the training set criterion  $L_T$  at  $(\hat{\theta}(\boldsymbol{\lambda}^{(k)}), \boldsymbol{\lambda}^{(k)})$ .

Let  $\tilde{\theta}(\boldsymbol{\lambda}^{(k)})$  be the optimal model parameters restricted to the space  $\Omega^{(k)}$

$$\tilde{\theta}(\boldsymbol{\lambda}^{(k)}) = \arg \min_{\theta \in \Omega^{(k)}} L_T(\theta, \boldsymbol{\lambda}^{(k)}) \quad (19)$$

Calculate the gradient

$$\nabla_{\boldsymbol{\lambda}} L(\mathbf{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\mathbf{X}_V))|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} = \left[ \frac{\partial}{\partial \theta} L(\mathbf{y}_V, f_{\theta}(\mathbf{X}_V))|_{\theta=\hat{\theta}(\boldsymbol{\lambda}^{(k)})} \right]^T \frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\theta}(\boldsymbol{\lambda})|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}}$$

Perform the gradient update:

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

**Example: Lasso** To better understood this procedure, we consider an example joint optimization problem with the lasso. At each iteration  $k$ , we solve the inner optimization problem

$$\hat{\beta}(\boldsymbol{\lambda}^{(k)}) = \arg \min_{\beta \in \mathbb{R}^p} \|\mathbf{y}_T - \mathbf{X}_T \beta\|_2^2 + \lambda^{(k)} \|\beta\|_1 \quad (20)$$

The differentiable space of this criterion at the point  $(\hat{\beta}(\boldsymbol{\lambda}^{(k)}), \boldsymbol{\lambda}^{(k)})$  is the space spanned by the non-zero elements of  $\hat{\beta}(\boldsymbol{\lambda}^{(k)})$  and  $\boldsymbol{\lambda}^{(k)}$ :

$$\Omega^{(k)} = \text{span}(\{e_i | \hat{\beta}(\boldsymbol{\lambda}^{(k)})_i \neq 0\} \cup e_{p+1}) \quad (21)$$

To get the gradient descent direction, define  $\tilde{\beta}(\boldsymbol{\lambda}^{(k)})$  to be the minimizer of the training loss over the space spanned by the non-zero coefficients of  $\hat{\beta}(\boldsymbol{\lambda}^{(k)})$ :

$$\tilde{\beta}(\boldsymbol{\lambda}^{(k)}) = \arg \min_{\beta: \beta_i=0 \text{ if } \hat{\beta}_i(\boldsymbol{\lambda}^{(k)})=0} \|\mathbf{y}_T - \mathbf{X}_T \beta\|_2^2 + \lambda^{(k)} \|\beta\|_1 \quad (22)$$

$\tilde{\beta}(\boldsymbol{\lambda}^{(k)})$  must satisfy the following gradient condition

$$0 = -\mathbf{X}_T^T (\mathbf{y}_T - \mathbf{X}_T \tilde{\beta}(\boldsymbol{\lambda}^{(k)})) + \lambda^{(k)} \text{sgn}(\tilde{\beta}(\boldsymbol{\lambda}^{(k)})) \quad (23)$$

Assuming that  $(\mathbf{X}_T^T \mathbf{X}_T)^{-1}$  exists, implicit differentiation of (23) gives us

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\beta}(\boldsymbol{\lambda})|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} = (\mathbf{X}_T^T \mathbf{X}_T)^{-1} \text{sgn}(\tilde{\beta}(\boldsymbol{\lambda}^{(k)})) \quad (24)$$

Note that we can ensure having an invertible matrix by adding a ridge penalty with a small regularization parameter to the training criterion. The gradient descent direction for the  $k$ th iteration is therefore

$$\nabla_{\boldsymbol{\lambda}} L(\mathbf{y}_V, f_{\hat{\theta}(\boldsymbol{\lambda})}(\mathbf{X}_V))|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(k)}} = (\mathbf{X}_V (\mathbf{X}_T^T \mathbf{X}_T)^{-1} \text{sgn}(\tilde{\beta}(\boldsymbol{\lambda}^{(k)})))^T (\mathbf{y}_V - \mathbf{X}_V \tilde{\beta}(\boldsymbol{\lambda}^{(k)})) \quad (25)$$

### 2.2.1 Variants of Gradient Descent

As shown later, joint optimization problems can be solved using variants of gradient descent, including those with decreasing or adaptive step sizes. In our simulation studies, we implemented both regular gradient descent and Nesterov's gradient descent with adaptive restarts. Nesterov's gradient descent is an accelerated method in which the step size depends on previous steps, and the momentum grows from one iteration to the next. Restarting refers to resetting the momentum back to zero. In our implementation of Nesterov's gradient descent with adaptive restarts, we restarted whenever the criterion increases. For more details, refer to the analysis by O'Donoghue and Candes.

## 2.3 Joint optimization for example regressions

In this subsection, we present example joint optimization problems and derivations for the gradient. We begin with the simple example of ridge regression, followed by the more complex examples elastic net, sparse group lasso, and additive partial linear models.

### 2.3.1 Ridge Regression

The joint optimization problem for ridge regression is:

$$\begin{aligned} & \min_{\lambda \in \mathbb{R}} \frac{1}{2} \|\mathbf{y}_V - \mathbf{X}_V \hat{\boldsymbol{\beta}}(\lambda)\|^2 \\ & \text{where } \hat{\boldsymbol{\beta}}(\lambda) = \arg \min_{\boldsymbol{\beta}} \frac{1}{2} \|\mathbf{y}_T - \mathbf{X}_T \boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \end{aligned} \quad (26)$$

Calculating the gradient is straightforward since there is a closed-form solution for  $\hat{\boldsymbol{\beta}}(\lambda)$ :

$$\hat{\boldsymbol{\beta}}(\lambda) = (\mathbf{X}_T^T \mathbf{X}_T + \lambda \mathbf{I})^{-1} \mathbf{X}_T^T \mathbf{y}_T \quad (27)$$

Differentiating the above equation with respect to  $\lambda$  and then using the chain rule gives us the gradient:

$$\nabla_{\lambda} L_V(\lambda) = (\mathbf{X}_V (\mathbf{X}_T^T \mathbf{X}_T + \lambda \mathbf{I})^{-1} \hat{\boldsymbol{\beta}})^T (\mathbf{y}_V - \mathbf{X}_V \hat{\boldsymbol{\beta}}(\lambda)) \quad (28)$$

### 2.3.2 Elastic Net

The elastic net combines the  $\ell_1$  and  $\ell_2$  penalties to address the limitations of lasso and ridge regression. Joint optimization finds the optimal regularization parameters  $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^T$  corresponding to each of these penalties:

$$\begin{aligned} & \min_{\boldsymbol{\lambda} \in \mathbb{R}^2} \frac{1}{2} \|\mathbf{y}_V - \mathbf{X}_V \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\|^2 \\ & \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg \min_{\boldsymbol{\beta}} \frac{1}{2} \|\mathbf{y}_T - \mathbf{X}_T \boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \frac{1}{2} \lambda_2 \|\boldsymbol{\beta}\|_2^2 \end{aligned} \quad (29)$$

By the KKT conditions, we have that:

$$0 \in -\mathbf{X}_T^T (\mathbf{y}_T - \mathbf{X}_T \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \lambda_1 s(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \lambda_2 \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \quad (30)$$

where  $s(\cdot)$  is the subdifferential of the lasso penalty and is defined by

$$s(\beta_i) = \begin{cases} \text{sign}(\beta_i) & \text{if } \beta_i \neq 0, \\ [-1, 1] & \text{otherwise} \end{cases} \quad (31)$$

We reduce the problem to just a gradient equation by restricting the features to the active set where  $\beta_i \neq 0$ . Let  $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$  be the active regression coefficients. In addition, let  $\dot{\mathbf{X}}_T$  and  $\dot{\mathbf{X}}_V$  be the training and validation set design matrices restricted to just the columns corresponding to the active set. The subgradient condition restricted to just the active regression coefficients is reduced to the following gradient condition:

$$-\dot{\mathbf{X}}_T^T(\mathbf{y}_T - \dot{\mathbf{X}}_T \dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) + \lambda_1 \text{sign}(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) + \lambda_2 \dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}) = 0 \quad (32)$$

Performing implicit differentiation gives us the gradient:

$$\nabla_{\boldsymbol{\lambda}} L_V(\boldsymbol{\lambda}) = \begin{bmatrix} (\dot{\mathbf{X}}_V^T (\dot{\mathbf{X}}_T^T \dot{\mathbf{X}}_T + \lambda_2 \mathbf{I})^{-1} \text{sign}(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})))^T (\mathbf{y}_V - \dot{\mathbf{X}}_V \dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \\ (\dot{\mathbf{X}}_V^T (\dot{\mathbf{X}}_T^T \dot{\mathbf{X}}_T + \lambda_2 \mathbf{I})^{-1} \dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}))^T (\mathbf{y}_V - \dot{\mathbf{X}}_V \dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \end{bmatrix} \quad (33)$$

### 2.3.3 Sparse Group Lasso

Sparse group lasso, a variant of the classical lasso regression, is intended for modeling problems with grouped covariates believed to have sparse effects on a group and a within-group level. Given  $m$  covariate groups, let the predictors  $\mathbf{X}$  and coefficient vector  $\boldsymbol{\beta}$  be split into submatrices  $\mathbf{X}^{(k)}$  and vectors  $\boldsymbol{\beta}^{(k)}$  for each covariate group  $k = 1, \dots, m$ . Sparse group lasso enforces group-level sparsity and within-group sparsity using the penalties  $\sum_{k=1}^m \|\boldsymbol{\beta}^{(k)}\|_2$  and  $\|\boldsymbol{\beta}\|_1$ . The joint optimization problem to get the optimal regularization parameters  $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^T$  is:

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}^2} \frac{1}{2n} \|\mathbf{y}_V - \mathbf{X}_V \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\|_2^2 \quad (34)$$

where  $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg \min_{\boldsymbol{\beta}} \frac{1}{2n} \|\mathbf{y}_T - \mathbf{X}_T \boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{k=1}^m \|\boldsymbol{\beta}^{(k)}\|_2 + \lambda_2 \|\boldsymbol{\beta}\|_1$

The KKT conditions state that

$$0 \in -\mathbf{X}_T^T(\mathbf{y}_T - \mathbf{X}_T \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \lambda_1 \sum_{l=1}^m t(\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})) + \lambda_2 s(\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})) \quad (35)$$

where  $s$  was defined previously. The subdifferential of the group lasso penalty is defined as

$$t(\boldsymbol{\beta}^{(k)}(\boldsymbol{\lambda})) = \begin{cases} \frac{\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})}{\|\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})\|_2} & \text{if } \boldsymbol{\beta}^{(k)} \neq \mathbf{0}, \\ \{t : \|t\|_2 \leq 1\} & \text{otherwise} \end{cases} \quad (36)$$

We reduce the problem by restricting to just the active set of features, which are those with nonzero coefficients  $\boldsymbol{\beta}_i^{(k)}$ . The reduced stationarity equation, in terms of the active regression coefficients  $\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})$  and restricted design matrices  $\dot{\mathbf{X}}_T$  and  $\dot{\mathbf{X}}_V$ , is

$$\nabla_{\boldsymbol{\lambda}} L_V(\boldsymbol{\lambda}) = \begin{bmatrix} \frac{1}{n} \left( \dot{\mathbf{X}}_V \left( \frac{1}{n} \dot{\mathbf{X}}_T^T \dot{\mathbf{X}}_T - \lambda_1 \mathbf{M}_1 + \lambda_1 \mathbf{M}_2 \right)^{-1} t(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \right)^T (\mathbf{y}_V - \dot{\mathbf{X}}_V \dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \\ \frac{1}{n} \left( \dot{\mathbf{X}}_V \left( \frac{1}{n} \dot{\mathbf{X}}_T^T \dot{\mathbf{X}}_T - \lambda_1 \mathbf{M}_1 + \lambda_1 \mathbf{M}_2 \right)^{-1} s(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \right)^T (\mathbf{y}_V - \dot{\mathbf{X}}_V \dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \end{bmatrix} \quad (37)$$

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

### 2.3.4 Additive Partial Linear Models

Additive partial linear models (APLMs), a semi-parametric regression model, is used when the response is believed to depend on some of the predictors linearly and the other predictors nonlinearly. More specifically, given the predictors  $\mathbf{x} \in \mathbb{R}^n$  and  $z \in \mathbb{R}$ , the predicted response  $y$  is  $\mathbf{x}^T \boldsymbol{\beta} + g(z)$ , where  $\boldsymbol{\beta}$  is the regression coefficients and  $g$  is some nonlinear univariate function.

For this example, the APLM uses a lasso penalty for the linear component and an  $\ell_2$  trend-filtering penalty for the nonlinear component. Let the data set have linear predictors  $\mathbf{X} \in \mathbb{R}^{n \times p}$  and nonlinear predictors  $\mathbf{z} \in \mathbb{R}^n$ . We assume that the observations are ordered by increasing  $z_i$ . The training set  $\mathbf{X}_T$  is randomly chosen from the data and can be obtained using matrix multiplication  $\mathbf{X}_T = \mathbf{M}_T \mathbf{X}$  for



some matrix  $\mathbf{M}_T$ . The validation set  $\mathbf{X}_V$ , composed of the remaining observations, are obtained using a similarly defined matrix  $\mathbf{M}_V$ . The joint optimization problem to obtain the optimal penalty parameters  $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^T$  is as follows:

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}^2} \frac{1}{2} \|\mathbf{y}_V - \mathbf{X}_V \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \mathbf{M}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\|_2^2 \quad (38)$$

where  $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg \min_{\boldsymbol{\beta}, \boldsymbol{\theta}} \frac{1}{2} \|\mathbf{y}_T - \mathbf{X}_T \boldsymbol{\beta} - \mathbf{M}_T \boldsymbol{\theta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \frac{1}{2} \lambda_2 \|\mathbf{D} \boldsymbol{\theta}\|_2^2$

$\mathbf{D}$  is the second-order difference matrix for  $\mathbf{z}$ :

$$\mathbf{D} = \mathbf{D}^{(1)} \cdot \text{diag}\left(\frac{1}{z_2 - z_1}, \frac{1}{z_3 - z_2}, \dots, \frac{1}{z_n - z_{n-1}}, 0\right) \cdot \mathbf{D}^{(1)} \quad (39)$$

where

$$\mathbf{D}^{(1)} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n} \quad (40)$$

The KKT conditions state that:

$$\begin{aligned} \mathbf{0} &\in -\mathbf{X}_T^T (\mathbf{y}_T - \mathbf{X}_T \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \mathbf{M}_T \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) + \lambda_1 s(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \\ \mathbf{0} &= -\mathbf{M}_T^T (\mathbf{y}_T - \mathbf{X}_T \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \mathbf{M}_T \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) + \lambda_2 \mathbf{D} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{aligned} \quad (41)$$

We reduce first equation to an implicit equation involving gradients. We restrict the problem to involve just the active set of features that correspond to nonzero  $\beta_i$ . Let the active regression coefficients be  $\dot{\boldsymbol{\beta}}(\boldsymbol{\lambda})$  and restricted design matrices be  $\dot{\mathbf{X}}_T$  and  $\dot{\mathbf{X}}_V$ . The subgradient condition is therefore reduced to the following equation:

$$\mathbf{0} = -\dot{\mathbf{X}}_T^T (\mathbf{y}_T - \dot{\mathbf{X}}_T \dot{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \mathbf{M}_T \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) + \lambda_1 s(\dot{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \quad (42)$$

By the chain rule, the gradient is:

$$\nabla_{\boldsymbol{\lambda}} L_V(\boldsymbol{\lambda})_i = -\left( \dot{\mathbf{X}}_V \frac{\partial \dot{\boldsymbol{\beta}}(\boldsymbol{\lambda})}{\partial \lambda_i} + \mathbf{M}_V \frac{\partial \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}{\partial \lambda_i} \right)^T \left( \mathbf{y}_V - \dot{\mathbf{X}}_V \dot{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \mathbf{M}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right) \text{ for } i = 1, 2 \quad (43)$$

The partial derivatives are obtained from implicit differentiation of the reduced KKT conditions.

$$\begin{aligned} \frac{\partial}{\partial \lambda_1} \dot{\boldsymbol{\beta}}(\boldsymbol{\lambda}) &= -\left( \dot{\mathbf{X}}_T^T \dot{\mathbf{X}}_T - \dot{\mathbf{X}}_T^T \mathbf{M}_T (\mathbf{M}_T^T \mathbf{M}_T + \lambda_2 \mathbf{D}^T \mathbf{D})^{-1} \mathbf{M}_T^T \dot{\mathbf{X}}_T \right)^{-1} s(\dot{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \\ \frac{\partial}{\partial \lambda_1} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) &= -\left( \mathbf{M}_T^T \mathbf{M}_T + \lambda_2 \mathbf{D}^T \mathbf{D} \right)^{-1} \mathbf{M}_T^T \dot{\mathbf{X}}_T \frac{\partial}{\partial \lambda_1} \dot{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \lambda_2} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) &= -\left( \mathbf{M}_T^T \mathbf{M}_T - \mathbf{M}_T^T \dot{\mathbf{X}}_T (\dot{\mathbf{X}}_T^T \dot{\mathbf{X}}_T)^{-1} \dot{\mathbf{X}}_T^T \mathbf{M}_T + \lambda_2 \mathbf{D}^T \mathbf{D} \right) \mathbf{D}^T \mathbf{D} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial \lambda_2} \dot{\boldsymbol{\beta}}(\boldsymbol{\lambda}) &= -\left( \dot{\mathbf{X}}_T^T \dot{\mathbf{X}}_T \right)^{-1} \dot{\mathbf{X}}_T^T \mathbf{M}_T \frac{\partial}{\partial \lambda_2} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{aligned} \quad (44)$$

### 3 Results: validation error minimization

The purpose of this simulation study is to compare the performance and efficiency of descent-based and grid-based joint optimization in minimizing the validation loss for different regularization methods. Experiments were run for the example regressions elastic net, sparse group lasso, and APLMs. For each experiment, we implemented three methods to tune the regularization parameters: cross validation over a grid of parameter values, gradient descent with a constant step size, and Nesterov's gradient descent with adaptive restarts. Below, we describe the simulation settings, followed by a discussion of the results.

### 3.1 Elastic net

We generated 30 datasets with 80 training and 20 validation observations each and 250 predictors. The predictors were randomly generated such that the pairwise correlation between predictors  $x_i$  and  $x_j$  was  $\text{corr}(i, j) = 0.5^{|i-j|}$ . The response vector  $y$  was constructed as

$$y = X\beta + \sigma\epsilon \quad (45)$$

where the first 15 values of the coefficient vector  $\beta$  are ones and the rest are zeroes and  $\epsilon \sim N(0, 1)$ .  $\sigma$  was chosen such that the signal to noise ratio is 2.

Both gradient descents were initialized at (0.01, 0.01) and (10, 10). The grid for cross validation was  $10 \times 10$ , starting 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 iid gaussian. The response  $y$  was constructed as

$$y = \sum_{l=1}^3 X^{(k)} \beta^{(k)} + \sigma\epsilon \quad (46)$$

where  $\epsilon \sim N(0, 1)$ ,  $\beta^{(k)} = (1, 2, 3, 4, 5, 0, \dots, 0)$  for  $k = 1, 2, 3$ .  $\sigma$  was chosen such that the signal to noise ratio was 2. The number of groups  $m$  in the criterion was set to 150.

Both gradient descents were initialized at three points, where all regularization parameters were set to 0.01, 1, and 100. We used a  $10 \times 10$  grid for grid-based cross validation, starting from  $1e-5$  to  $\max(\{\|X^{(k)T} y\|_2\}_{l=1, \dots, m})$ .

### 3.3 Additive partial linear models

We generated 30 datasets with 100 training and 25 validation observations and 21 predictors. The first 20 predictors were the inputs to the linear function and the last predictor was the input to the nonlinear function. Predictors for the linear function were generated such that the first two groups of three features were highly correlated and the rest of the features were generated iid Gaussian.

$$\begin{aligned} x_i &= Z_1 + \epsilon_i \text{ for } i = 1, 2, 3 \\ x_i &= Z_2 + \epsilon_i \text{ for } i = 4, 5, 6 \\ x_i &\sim N(0, 1) \text{ for } i = 7, \dots, 20 \end{aligned} \quad (47)$$

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

$$y = X\beta + \kappa g(z) + \sigma\epsilon \quad (48)$$

The constants  $\kappa$  and  $\sigma$  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, respectively. We set  $\beta = (1, 1, 1, 1, 1, 1, 0, \dots, 0)$  and  $g(z) = (2 - z) \sin(20z^4)$ .

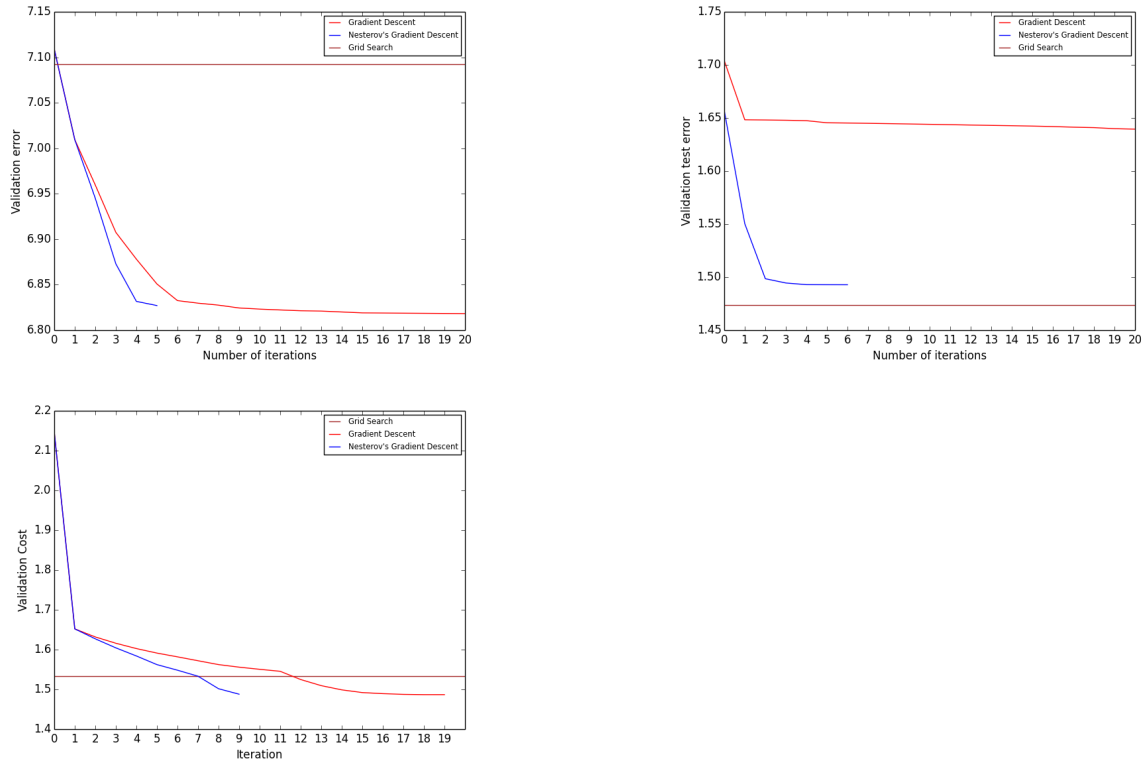
Both gradient descents were initialized at the four points  $\lambda_1 = \lambda_2 = 10^i$  for  $i = -2, -1, 0, 1$ . We used a  $10 \times 10$  grid for grid-based cross validation, with values ranging from  $1e - 6$  to 10.

### 3.4 Discussion of results

As shown in Table 1, descent-based and grid-based joint optimization had the same performance in all three situations. The only major difference between the different implementations of joint optimization is in the runtimes. In every situation, gradient descent was over 40% faster than the grid-based approach and Nesterov's subgradient descent was over 50% faster.

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 1.** Validation Error comparisons



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

Figure 2 shows example plots of how validation error decreases in the descent-based joint optimization procedures. The horizontal line in the plots are the lowest validation error achieved by grid-based joint optimization. In all three plots, Nesterov’s gradient descent requires fewer iterations to minimize the validation loss.

## 4 Results: regularizations with more than two penalties

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

For each experiment, given a regularization method, we compared tuning the original two-parameter version using a grid-based approach to tuning the generalized, many-parameter version using gradient descent. In the sparse group lasso experiment, we tested an “unpooled” version of sparse group lasso in which the number of regularization parameters is linear in the number of covariate groups. In the APLM experiment, we tested having an additional ridge penalty for the linear regression coefficients. To compare the regularization methods, we measured the loss on the test sets. As shown later, since gradient descent for the generalized regression achieved a lower test set loss on average, we have shown that a descent-based approach for many-penalty regressions can produce better models than those from 2-penalty regressions.

The joint optimization problem formulations and gradient derivations for these generalized regularizations are given in the Appendix.

### 4.1 Unpooled sparse group lasso

In this experiment, we compared the effectiveness of applying gradient descent on an “unpooled” variant of sparse group lasso to grid-based joint optimization on the regular sparse group lasso. In unpooled sparse group lasso, the group lasso penalty parameter  $\lambda_1$  is replaced with an individual parameter for each covariate group:

$$\sum_{l=1}^m \lambda_1^{(k)} \|\beta^{(k)}\|_2 + \lambda_2 \|\beta\|_1 \quad (49)$$

Hence, the number of regularization parameters increased from two to one plus the number of coefficient groups. This unpooled variant allows different thresholds to be used for setting coefficients and coefficient groups to zero, as opposed to the original sparse group lasso where the thresholds are shared. Potentially, this increased flexibility can produce more accurate models, especially when the covariate groups have very different distributions.

We ran three experiments, each with a different number of covariate groups  $m$  and total covariates  $p$ . The simulation settings were similar to the first simulation study. Grid-based joint optimization for sparse group lasso was initialized with the same grid. Gradient descent for unpooled sparse group lasso was initialized with a lower set of values 1e-4, 1e-3, and 1e-2 to compensate for the fact that there are now  $m + 1$  regularization parameters.

We measured three metrics to assess model performance: test error,  $\beta$  error, which is defined as  $\|\beta - \hat{\beta}\|_2^2$ , and the percentage of nonzero coefficients correctly identified among all the true nonzero coefficients. As shown in Table 2, unpooled sparse group lasso performed better according to all three metrics in all the experiments. Furthermore, gradient descent for unpooled sparse group lasso is faster than grid-based joint optimization for two-parameter sparse group lasso, even in the case with 151 regularization parameters.

n=60, p=300, g=3, m=30				
	$\beta$ Error	% correct nonzero $\beta$	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				
	$\beta$ Error	% correct nonzero $\beta$	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				
	$\beta$ Error	% correct nonzero $\beta$	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 2.** Unpooled sparse group lasso

## 4.2 Additive partial linear models with three regularization parameters

In this example of joint optimization for APLMs, we experimented with a generalized version of the APLM regularization by using the elastic net instead of the lasso for the linear regression coefficients:

$$\lambda_1 \|\beta\|_1 + \frac{1}{2} \lambda_2 \|\beta\|_2^2 + \frac{1}{2} \lambda_3 \|D^{(2)}\theta\|_2^2 \quad (50)$$

We compared model fits from this three-parameter regularization tuned using gradient descent to those from the original two-parameter regularization tuned using a grid-based approach. Our hypothesis was that the elastic net penalty in this generalized regularization would produce more accurate models since it encourages both sparsity of representation and grouping among strongly correlated predictors.

We ran experiments on three nonlinear functions of varying levels of smoothness. The simulation settings were similar to before. The only difference was that gradient descent was initialized at the six points  $\lambda_1 = \lambda_2 = \lambda_3 = 10^i$  for  $i = -4, \dots, 1$ .

Models were judged by their predictive accuracy. In addition, we used  $\beta$  error, defined as  $\|\beta - \hat{\beta}\|_2^2$ , and  $\theta$  error, defined as  $\|g(z) - \theta\|_2^2$ , to measure the error in the estimated linear and nonlinear components, respectively.

The results in Table 3 show that using three penalty terms in the criterion improved the models by all metrics. The biggest improvement was in the estimation of the linear regression components, which is expected since we added an penalty term for the linear coefficients. Somewhat surprisingly, the estimation of the nonlinear components also improved slightly, even though no penalty term was added for the nonlinear component. The runtime for tuning the three-parameter regularization was slightly longer than the grid-based version, but remained reasonable nonetheless. Most importantly, the increase in runtime is not exponential in the number of penalty parameters.

Figure 3 shows 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.

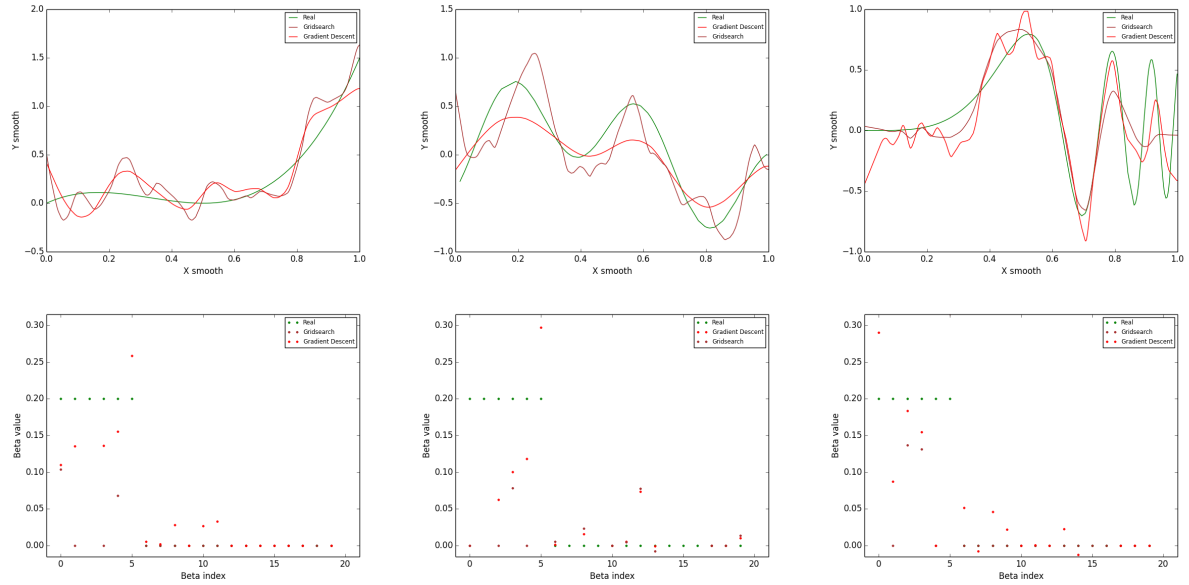
## 5 Application to Real Data

In this section, we discuss the performance and efficiency of joint optimization when applied to real data. We use un-pooled sparse group lasso as our example, since it can have the most number of penalty parameters among the regularization methods we previously mentioned. We compare this to sparse group lasso tuned using grid search.

We run regression to predict disease phenotype from gene expression data, as presented by Noah et. al. (2007). In this problem, the genes are grouped by gene pathways and we are interested in the pathways of interest and, from them, the driving genes.

$g(z) = 4z^3 - z^2 + 2z$				
	$\beta$ Error	$\theta$ error	Test Error	Runtime (sec)
Gridsearch	0.59	3.35	3.78	35.48
Gradient Descent	0.38	2.96	3.73	43.44
$g(z) = \sin(5z) + \sin(15(z - 3))$				
	$\beta$ Error	$\theta$ error	Test Error	Runtime (sec)
Gridsearch	0.51	3.76	3.90	37.04
Gradient Descent	0.34	3.73	3.79	45.95
$g(z) = (2 - z) \sin(20z^4)$				
	$\beta$ Error	$\theta$ 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 3.** Additive Partial Linear Model



**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 4.** Ulcerative Colitis Data: SGL = sparse group lasso, variance in parentheses

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

Perhaps the most important result is that joint optimization makes it possible to compare un-pooled sparse group lasso to sparse group lasso. Un-pooled sparse group lasso requires 327 regularization parameters for this problem, which is impossible to tune using grid search. Treating the problem as a continuous optimization problem not only makes it possible to tune many penalty parameters, but also makes the runtime comparable to grid search over two parameters, as seen in Table 4.

Un-pooled sparse group lasso and sparse group lasso achieve similar classification rates around 83%. The primary difference is that 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. The number of genesets identified by un-pooled sparse group lasso ranged from 8 to 12. Consistency is certainly an useful quality in regularization methods. These results suggest that un-pooling the penalty parameters in sparse group lasso can improve the regularization method. Further research is needed to determine if un-pooled sparse group lasso is a useful regularization method.

In this example, we have demonstrated that joint optimization is highly efficient at tuning parameters and can produce meaningful solutions to regression problems for real data.

## 6 Discussion

We proposed using joint optimization for tuning regularization parameters in both linear and nonlinear regressions. By treating the regression as an optimization problem over the regularization parameter space, joint optimization is highly scalable in the number of parameters. Through experiments on simulated data, we demonstrated that joint optimization is able to fit models with validation errors as low as grid search. We then increased the number of regularization parameters in traditional regularization methods and used joint optimization to fit even better models. During this exploration, we discovered that ”unpooled” sparse group lasso has significantly better performance than the traditional sparse group lasso. Further research should be done to explore this new regularization method. Finally, we demonstrated that this method works well on real data.

This paper applied joint optimization to a number of criteria, but this method should be tested on more criteria to analyze its robustness. We also expect that by implementing joint optimization with more sophisticated optimization methods, we can expect significant gains in efficiency. Finally, it would be worthwhile to understand what types of situations are ill-suited for joint optimization and what is the maximum number of regularization parameters joint optimization can tune with reasonable computation time.

# Appendix

## 6.1 Proof for Theorem 1

*Proof.* It is sufficient to show that the theorem holds for a given  $\lambda_0$ . For a given  $\lambda_0$ , 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  $\theta$  with respect to the basis  $B$ :

$$\tilde{L}(\theta, \lambda) = \nabla_B L_T(\cdot, \lambda)|_{\theta} \quad (51)$$

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  $\hat{\theta}(\lambda_0)$ :

$$\nabla_B L_T(\cdot, \lambda_0)|_{\hat{\theta}(\lambda_0)} = \tilde{L}_T(\hat{\theta}(\lambda_0), \lambda_0) = 0 \quad (52)$$

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  $\lambda_0$  and  $V$  containing  $\hat{\theta}(\lambda_0)$  and a continuously differentiable function  $\gamma : U \rightarrow V$  such that for every  $\lambda \in U$ , we have that

$$\tilde{L}_T(\gamma(\lambda), \lambda) = \nabla_B L_T(\cdot, \lambda)|_{\gamma(\lambda)} = 0 \quad (53)$$

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  $\lambda_0$  such that for every  $\lambda \in W'$ ,

$$\hat{\theta}(\lambda) = \arg \min_{\theta} L_T(\theta, \lambda) = \arg \min_{\theta \in \Omega^{L_T}(\hat{\theta}(\lambda_0), \lambda_0)} L_T(\theta, \lambda) = \gamma(\lambda) \quad (54)$$

Therefore, we have shown that if  $\lambda_0$  satisfies the assumptions given in the theorem, the fitted model parameters  $\hat{\theta}(\lambda)$  is a continuously differentiable function within a neighborhood of  $\lambda_0$ . 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

$$\frac{\partial}{\partial \lambda} \hat{\theta}(\lambda) = \frac{\partial}{\partial \lambda} \dot{\theta}(\lambda) \quad (55)$$

where

$$\dot{\theta}(\lambda) = \arg \min_{\theta \in \Omega^{L_T}(\hat{\theta}(\lambda_0), \lambda_0)} L_T(\theta, \lambda) \quad (56)$$

We can then apply the chain rule to get the gradient of the validation loss.  $\square$

## 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:

$$\begin{aligned}\nabla_{\lambda_1^{(k)}} L(\mathbf{y}_V, X_V \hat{\beta}) &= \frac{1}{n} \left( X_V \left( \frac{1}{n} X^T X + M_1 + M_2 \right)^{-1} \begin{bmatrix} 0 \\ \vdots \\ \frac{\hat{\beta}^{(l)}}{\|\hat{\beta}^{(l)}\|_2} \\ \vdots \\ 0 \end{bmatrix} \right)^T (\mathbf{y}_V - X_V \hat{\beta}) \\ \nabla_{\lambda_2} L(\mathbf{y}_V, X_V \hat{\beta}) &= \frac{1}{n} \left( \left( \frac{1}{n} X^T X + M_1 + M_2 \right)^{-1} \text{sign}(\hat{\beta}) \right)^T (\mathbf{y}_V - X_V \hat{\beta})\end{aligned}\quad (57)$$

where  $M_1$  is the block diagonal matrix with  $l$  components

$$\frac{\lambda_1^{(k)}}{\|\beta^{(k)}\|_2^3} \begin{bmatrix} \beta_1^{(k)} & 0 \\ 0 & \beta_2^{(k)} \\ & \ddots \end{bmatrix} \begin{bmatrix} - & \beta^{(k)T} & - \\ - & \beta^{(k)T} & - \\ \vdots & & \end{bmatrix} \text{ from top left to bottom right and } M_2 \text{ is the diagonal matrix}$$

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:

$$\begin{aligned}\min_{\lambda_1, \lambda_2, \lambda_3} \frac{1}{2} \|\mathbf{y}_V - X_V \hat{\beta} - M_V \hat{\theta}\|_2^2 \\ \text{where } \hat{\beta}, \hat{\theta} = \arg \min_{\beta, \theta} \frac{1}{2} \|y - X\beta - M_T \theta\|_2^2 + \lambda_1 \|\beta\|_1 + \frac{1}{2} \lambda_2 \|\beta\|_2^2 + \frac{1}{2} \lambda_3 \|D^{(2)} \theta\|_2^2\end{aligned}\quad (58)$$

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:

$$\nabla_{\lambda_i} L(\mathbf{y}_V, X_V \hat{\beta} + \hat{\theta}) = (X_V \frac{\partial \beta}{\partial \lambda_i} + M_V \frac{\partial \theta}{\partial \lambda_i})^T (\mathbf{y}_V - X_V \hat{\beta} - \hat{\theta}_V) \quad (59)$$

where

$$\begin{aligned}\frac{\partial \beta}{\partial \lambda_1} &= -(X_T^T X_T - X_T^T M_T (M_T^T M_T + \lambda_3 D^T D)^{-1} M_T^T X_T + \lambda_2 I)^{-1} \text{sign}(\hat{\beta}) \\ \frac{\partial \theta}{\partial \lambda_1} &= -(M_T^T M_T + \lambda_3 D^T D)^{-1} M_T^T X_T \frac{\partial \beta}{\partial \lambda_1} \\ \frac{\partial \beta}{\partial \lambda_2} &= -(X_T^T X_T - X_T^T M_T (M_T^T M_T + \lambda_3 D^T D)^{-1} M_T^T X_T + \lambda_2 I)^{-1} \hat{\beta} \\ \frac{\partial \theta}{\partial \lambda_2} &= -(M_T^T M_T + \lambda_3 D^T D)^{-1} M_T^T X_T \frac{\partial \beta}{\partial \lambda_2} \\ \frac{\partial \theta}{\partial \lambda_3} &= -(M_T^T M_T - M_T^T X_T (X_T^T X_T + \lambda_2 I)^{-1} X_T^T M_T + \lambda_3 D^T D)^{-1} D^T D \theta \\ \frac{\partial \beta}{\partial \lambda_3} &= -(X_T^T X_T + \lambda_2 I)^{-1} X_T^T M_T \frac{\partial \theta}{\partial \lambda_3}\end{aligned}\quad (60)$$

## Acknowledgments

BLAH

---

## References

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