Descent-based joint optimization

Jean Feng and Noah Simon

Abstract

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

1 Introduction

Consider the usual regression framework with p features and a response measured on n observations. Let X denote the $n \times p$ design matrix and y the response vector. For ill-posed or high-dimensional problems ($p \gg n$), there can often be an infinite number of solutions that minimize the loss function L but have high generalization error. A common solution is to use regularization, or penalization, to select models with desirable properties, such as smoothness and sparsity.

In recent years, there has been much interest in combining regularization methods to produce models with multiple desired characteristics. Examples include the elastic net (Zou and Hastie 2003), which combines the lasso and ridge penalties, and the sparse group lasso (Simon 2007), which combines the group lasso and lasso penalties. The general form of these regression problems is:

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

where L is the loss function, $\{J_i\}_{i=1,...,k}$ are the penalty functions, and $\{\lambda_i\}_{i=1,...,k}$ are the regularization parameters.

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

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

The simplest approach to solving the problem above is brute force: one fits models over a grid of parameter values and selects the model with the lowest validation error. As long as the grid is large and fine enough, this method of "grid search" will find a solution close to the global optimum. This approach is the current standard for choosing penalty parameters via train/validation. Unfortunately, it is

computationally intractable in cases with more than two parameters. Many variants of grid search have been proposed to increase efficiency, but their runtimes are all exponential in the number of parameters.

In this paper, we propose leveraging the tools of optimization to solve (2) over the penalty parameter space. We give a gradient descent algorithm for the penalty parameters (to minimize validation error). In contrast to an exhaustive "grid search", this "descent-based" optimization makes use of the smoothness of our validation-error surface.

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

Lorbert and Ramadge (2010) presented some related work on this topic. They solved linear regression problems by updating regression coefficients and regularization parameters using cyclical coordinate gradient descent. We take a more general approach that allows us to apply joint optimization to both linear and nonlinear regressions. In particular this paper focuses on three examples that demonstrate the wide applicability of our method: elastic net, sparse group lasso, and additive partial linear models.

In Section 2, we describe descent-based optimization in detail and present an algorithm for solving it in example regressions. In Section 3, we show that our method achieves validation errors as low as those achieved by grid search. In Section 4, we explore variants of the example regression problems that have many more regularization parameters and demonstrate that solving (2) is still computationally tractable. Finally, we present results on data predicting colitis status from gene expression in Section 5.

2 Descent-based Joint Optimization

2.1 Definition

Suppose that we have n observations on each of which we have measured p features and a response. Let $\boldsymbol{y}$ be the n response vector and $\boldsymbol{X}$ be the $n \times p$ design matrix. In addition, suppose we are fitting a model f by minimizing a penalized criterion of the form:

$$L(\boldsymbol{y}, f(\boldsymbol{X})) + \sum_{i=1}^{k} \lambda_i J_i(f) \tag{3}$$

where L is the loss function and (J_i, λ_i) is the ith penalty function and the corresponding penalty parameter.

Suppose we would to select penalty parameters $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_k)^T$ via a training/validation split. We partition the data into a training set $(\boldsymbol{y}_T, \boldsymbol{X}_T)$ and validation set $(\boldsymbol{y}_V, \boldsymbol{X}_V)$. We choose our penalty parameters to minimize the validation error by solving the following optimization problem:

$$\begin{gathered} \arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^k} L(\boldsymbol{y}_V, \hat{f}(\boldsymbol{X}_V|\boldsymbol{\lambda})) \\ \text{where } \hat{f}(\cdot|\boldsymbol{\lambda}) = \arg\min_f L(\boldsymbol{y}_T, f(\boldsymbol{X}_T)) + \sum_{i=1}^{k} \lambda_i J_i(f) \end{gathered} \tag{4}$$

For ease of reading, we will use $L_V(\boldsymbol{\lambda})$ to denote $L(\boldsymbol{y}_V, \hat{f}(X_V|\boldsymbol{\lambda}))$ since the validation loss is ultimately a function of the penalty parameters.

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

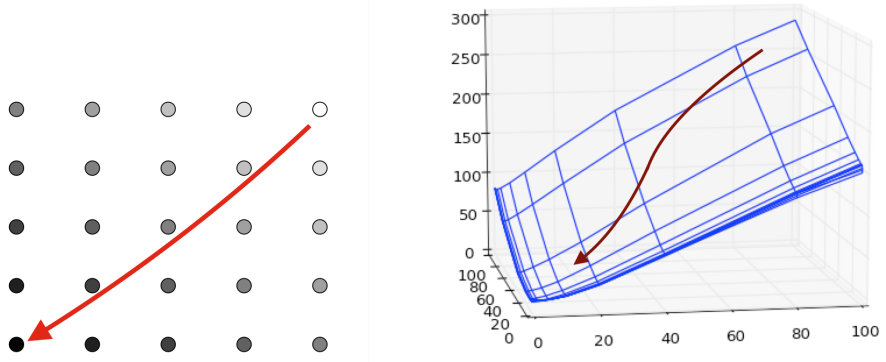

Figure 1. Left: darker points mean lower validation loss. Descent-based optimization descends in the most direct path towards the point producing the lowest validation loss. Right: The 3D version. We can tune regularization parameters using a grid search... or just descend opposite of the gradient.

search is inefficient (and becomes quickly completely intractable). In addition (4) is generally a continuous, piecewise-smooth problem. Using an exhaustive search ignores information available from the smoothness of the surface.

We propose leveraging the tools of smooth optimization to solve (4). In particular we discuss iterative methods, based on walking in a descent direction until convergence to a local minimum. In the simple case where the criterion is differentiable with respect to the penalty parameters, it is straightforward to use gradient descent or some variant thereof. We show that, with some slight tweaks, gradient descent can be applied in situations where the penalty is only differentiable when restricted to directions involving 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 Gradient Descent for Joint Optimization

In this section, we show how joint optimization problems can be solved using gradient descent. We first explain this method for the simple case where the validation loss $L_V(\boldsymbol{\lambda})$ is differentiable with respect to $\boldsymbol{\lambda}$. We then show that for cases where the validation loss is non-differentiable, we can reduce the problem and still solve it using gradient descent. Finally, we discuss variants of gradient descent that can also be used.

2.2.1 Differentiable functions! The simple case!

Suppose $L_V(\boldsymbol{\lambda})$ is differentiable everywhere in $\boldsymbol{\lambda}$. We can solve the joint optimization using gradient descent, as detailed in the box below:

Choose step size α.
Initialize $\lambda^{(0)}$.
For each iteration k:

$$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - \alpha \nabla L_V(\boldsymbol{\lambda}^{(k)})$$

The complexity in the algorithm above is hidden in the gradient term $\nabla L_V(\boldsymbol{\lambda})$. By the chain rule, the gradient requires calculating the gradient of the minimizer of penalized regression $\hat{f}$:

$$\nabla_\lambda L_V(\boldsymbol{\lambda}) = \frac{\partial}{\partial \hat{f}} L_V(\hat{f}(\boldsymbol{\lambda}))^T \frac{\partial}{\partial \lambda} \hat{f}(\boldsymbol{\lambda}) \tag{5}$$

If there is a closed-form solution for $\hat{f}(\cdot|\lambda)$, this calculation is straightforward. Ridge regression is an example of this case.

However, there is usually no closed-form solution. Instead, we rely on the KKT conditions for an implicit equation of the minimizer of the penalized regression. The stationarity condition states that, at optimality, the gradient of the penalized criterion with respect to $\hat{f}$ is zero:

$$0 = \nabla_{\hat{f}} \left(L(\boldsymbol{y}_T, \hat{f}(X_T)) + \sum_{i=1}^{k} \lambda_i J_i(\hat{f}) \right) \tag{6}$$

We then perform implicit differentiation of the equation above with respect to $\boldsymbol{\lambda}$ to get the gradient $\frac{\partial \hat{f}}{\partial \boldsymbol{\lambda}}$. This can then be plugged into equation 5 to get the gradient $\nabla L_V(\boldsymbol{\lambda}^{(k)})$ to perform gradient descent.

Note that the gradient $\frac{\partial \hat{f}}{\partial \boldsymbol{\lambda}}$ is often in terms of $\hat{f}$, which means that each gradient step requires solving the "inner" optimization problem. While this may seem costly, using warm starts significantly reduces the number of iterations needed to solve the inner problem.

2.2.2 Differentiable almost everywhere

For penalty functions that aren't differentiable everywhere, such as the lasso, we cannot directly apply gradient descent to perform joint optimization. However, if certain assumptions are satisfied, we can perform a modified gradient descent, which relies on the concept of an "active set." We define the "active set" of the model parameters f, denoted as $\dot{f}$, as the elements f_i where $L(f, \lambda)$ is twice continuously differentiable with respect to f_i. For example, in lasso regression, the active set is the nonzero regression coefficients.

Our modified gradient descent relies on two assumptions: one, the penalty function is twice continuously differentiable almost everywhere and, two, elements not in the active set remain locally constant within a local neighborhood of λ for almost every λ. With these assumptions, we can perform gradient descent where the gradient at each iteration k is calculated with respect to the active set of $\hat{f}(\cdot|\boldsymbol{\lambda}^{(k)})$ instead:

$$\nabla_{\boldsymbol{\lambda}} L_V(\boldsymbol{\lambda}^{(k)}) = \frac{\partial}{\partial \dot{\hat{f}}} L_V(\dot{\hat{f}}(\boldsymbol{\lambda}^{(k)}))^T \frac{\partial}{\partial \boldsymbol{\lambda}} \dot{\hat{f}}(\boldsymbol{\lambda}^{(k)}) \tag{7}$$

We can prove that this modified gradient descent is equivalent to subgradient descent for the joint optimization problem. The following theorem follows directly from Lemma 1 and 2 in the appendix.

Theorem 1. *For the optimization problem in (4), suppose that $L(\boldsymbol{y}, f(\boldsymbol{X}))$ and $\sum_{i=1}^{k} \lambda_i J_i(f)$ are twice continuously differentiable with respect to f almost everywhere and that the hessian of*

$$L(\boldsymbol{y}_T, f(\boldsymbol{X}_T)) + \sum_{i=1}^{k} \lambda_i J_i(f) \tag{8}$$

with respect to f is nonsingular almost everywhere. In addition, suppose that for almost every $\boldsymbol{\lambda} \in \mathbb{R}^k$, the non-active set of $\hat{f}(\boldsymbol{\lambda})$ is locally constant within some local neighborhood. Then the objective function $L(\boldsymbol{y_V}, \hat{f}(\boldsymbol{X_V}|\boldsymbol{\lambda}))$ is continuously differentiable almost everywhere with respect to $\boldsymbol{\lambda}$ and the gradient, where it is defined, is

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y_V}, \hat{f}(\boldsymbol{X_V}|\boldsymbol{\lambda})) = \frac{\partial}{\partial \hat{f}} L_V(\dot{\hat{f}}(\boldsymbol{\lambda}))^T \frac{\partial}{\partial \boldsymbol{\lambda}} \dot{\hat{f}}(\boldsymbol{\lambda}) \tag{9}$$

All the example regressions in this paper satisfy the assumptions. One may note that lasso regression applied to high-dimensional datasets actually violates the requirement for the hessian to be nonsingular. However, one can simply add a ridge penalty with a tiny coefficient to make the hessian nonsingular.

2.2.3 Variants of Gradient Descent

Thus far, we have described using gradient descent with constant step size α, but decreasing or adaptive step sizes $\alpha(\boldsymbol{\lambda}^{(k)})$ can be as effective. 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'Donaghue 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:

$$\begin{gathered} \min_{\lambda \in \mathbb{R}} \tfrac{1}{2}||\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\beta}}(\lambda)||^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\lambda) = \arg\min_{\boldsymbol{\beta}} \tfrac{1}{2}||\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\beta}||^2 + \lambda ||\boldsymbol{\beta}||_2^2 \end{gathered} \tag{10}$$

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

$$\hat{\boldsymbol{\beta}}(\lambda) = (\boldsymbol{X}_T^T \boldsymbol{X}_T + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}_T^T \boldsymbol{y}_T \tag{11}$$

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

$$\nabla_\lambda L_V(\lambda) \quad = \quad (\boldsymbol{X}_V (\boldsymbol{X}_T^T \boldsymbol{X}_T + \lambda \boldsymbol{I})^{-1} \hat{\boldsymbol{\beta}})^T (\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\beta}}(\lambda)) \tag{12}$$

2.3.2 Elastic Net

The elastic net combines the ℓ_1 and ℓ_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:

$$\begin{gathered} \min_{\boldsymbol{\lambda} \in \mathbb{R}^2} \tfrac{1}{2}||\boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\beta}}(\lambda)||^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \tfrac{1}{2}||\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\beta}||^2 + \lambda_1 ||\boldsymbol{\beta}||_1 + \tfrac{1}{2}\lambda_2 ||\boldsymbol{\beta}||_2^2 \end{gathered} \tag{13}$$

By the KKT conditions, we have that:

$$0 \in -\boldsymbol{X}_T^T (\boldsymbol{y}_T - \boldsymbol{X}_T \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \lambda_1 s(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \lambda_2 \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \tag{14}$$

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

$$s(\beta_i) = \begin{cases} sign(\beta_i) & \text{if } \beta_i \neq 0, \\ [-1, 1] & \text{otherwise} \end{cases} \tag{15}$$

We reduce the problem to just a gradient equation by restricting the features to the active set where $\boldsymbol{\beta}_i \neq 0$. Let $\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})$ be the active regression coefficients. In addition, let $\dot{\boldsymbol{X}}_T$ and $\dot{\boldsymbol{X}}_V$ 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:

$$-\dot{\boldsymbol{X}}_T^T(\boldsymbol{y}_T - \dot{\boldsymbol{X}}_T\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) + \lambda_1 sign(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) + \lambda_2\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}) = 0 \tag{16}$$

Performing implicit differentiation gives us the gradient:

$$\nabla_\lambda L_V(\boldsymbol{\lambda}) = \begin{bmatrix} (\dot{\boldsymbol{X}}_V^T(\dot{\boldsymbol{X}}_T^T\dot{\boldsymbol{X}}_T + \lambda_2\boldsymbol{I})^{-1}sign(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})))^T(\boldsymbol{y}_V - \dot{\boldsymbol{X}}_V\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \\ (\dot{\boldsymbol{X}}_V^T(\dot{\boldsymbol{X}}_T^T\dot{\boldsymbol{X}}_T + \lambda_2\boldsymbol{I})^{-1}\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}))^T(\boldsymbol{y}_V - \dot{\boldsymbol{X}}_V\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \end{bmatrix} \tag{17}$$

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 $\boldsymbol{X}$ and coefficient vector $\boldsymbol{\beta}$ be split into submatrices $\boldsymbol{X}^{(k)}$ and vectors $\boldsymbol{\beta}^{(k)}$ for each covariate group $k = 1, ..., m$. 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:

$$\begin{gathered} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2} \frac{1}{2n}\left|\left|\boldsymbol{y}_V - X_V\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right|\right|_2^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\left|\left|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\beta}\right|\right|_2^2 + \lambda_1 \sum_{k=1}^m ||\boldsymbol{\beta}^{(k)}||_2 + \lambda_2||\boldsymbol{\beta}||_1 \end{gathered} \tag{18}$$

The KKT conditions state that

$$0 \in -\boldsymbol{X}_T^T(\boldsymbol{y}_T - \boldsymbol{X}_T\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) + \lambda_1 \sum_{l=1}^m t(\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})) + \lambda_2 s(\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})) \tag{19}$$

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

$$t(\boldsymbol{\beta}^{(k)}(\boldsymbol{\lambda})) = \begin{cases} \frac{\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})}{||\hat{\boldsymbol{\beta}}^{(k)}(\boldsymbol{\lambda})||_2} & \text{if } \boldsymbol{\beta}^{(k)} \neq \boldsymbol{0}, \\ \{t : ||t||_2 \leq 1\} & \text{otherwise} \end{cases} \tag{20}$$

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 $\dot{\hat{\boldsymbol{\beta}}}(\lambda)$ and restricted design matrices $\dot{\boldsymbol{X}}_T$ and $\dot{\boldsymbol{X}}_V$, is

$$\nabla_\lambda L_V(\boldsymbol{\lambda}) = \begin{bmatrix} \frac{1}{n}\left(\dot{\boldsymbol{X}}_V(\frac{1}{n}\dot{\boldsymbol{X}}_T^T\dot{\boldsymbol{X}}_T - \lambda_1\boldsymbol{M}_1 + \lambda_1\boldsymbol{M}_2)^{-1}t(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}))\right)^T(\boldsymbol{y}_V - \dot{\boldsymbol{X}}_V\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \\ \frac{1}{n}\left(\dot{\boldsymbol{X}}_V(\frac{1}{n}\dot{\boldsymbol{X}}_T^T\dot{\boldsymbol{X}}_T - \lambda_1\boldsymbol{M}_1 + \lambda_1\boldsymbol{M}_2)^{-1}s(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}))\right)^T(\boldsymbol{y}_V - \dot{\boldsymbol{X}}_V\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \end{bmatrix} \tag{21}$$

where $\boldsymbol{M}_1$ 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 $\boldsymbol{M}_2$ is the block diagonal matrix with components $||\dot{\hat{\boldsymbol{\beta}}}^{(k)}(\boldsymbol{\lambda})||_2^{-1}\boldsymbol{I}$ for $k = 1, ..., m$ 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 $\boldsymbol{x} \in \mathbb{R}^n$ and $z \in \mathbb{R}$, the predicted response y is $\boldsymbol{x}^T\boldsymbol{\beta} + g(z)$, 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 ℓ_2 trend-filtering penalty for the nonlinear component. Let the data set have linear predictors $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ and nonlinear predictors $\boldsymbol{z} \in \mathbb{R}^n$. We assume that the observations are ordered by increasing z_i. The training set $\boldsymbol{X}_T$ is randomly chosen from the data and can be obtained using matrix multiplication $\boldsymbol{X}_T = \boldsymbol{M}_T\boldsymbol{X}$ for some matrix $\boldsymbol{M}_T$. The validation set $\boldsymbol{X}_V$, composed of the remaining observations, are obtained using a similarly defined matrix $\boldsymbol{M}_V$. The joint optimization problem to obtain the optimal penalty parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^T$ is as follows:

$$\begin{array}{c} \min_{\boldsymbol{\lambda}\in\mathbb{R}^2} \frac{1}{2}\left|\left|\boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \boldsymbol{M}_V\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\right|\right|_2^2 \\ \text{where } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}), \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta},\boldsymbol{\theta}} \frac{1}{2}\left|\left|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\beta} - \boldsymbol{M}_T\boldsymbol{\theta}\right|\right|_2^2 + \lambda_1||\boldsymbol{\beta}||_1 + \frac{1}{2}\lambda_2||\boldsymbol{D}\boldsymbol{\theta}||_2^2 \end{array} \tag{22}$$

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

$$\boldsymbol{D} = \boldsymbol{D}^{(1)} \cdot \text{diag}\big(\frac{1}{z_2 - z_1}, \frac{1}{z_3 - z_2}, ..., \frac{1}{z_n - z_{n-1}}, 0\big) \cdot \boldsymbol{D}^{(1)} \tag{23}$$

where

$$\boldsymbol{D}^{(1)} = \begin{bmatrix} -1 & 1 & 0 & ... & 0 & 0 \\ 0 & -1 & 1 & ... & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & ... & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{24}$$

The KKT conditions state that:

$$\begin{array}{rcl} \boldsymbol{0} & \in & -\boldsymbol{X}_T^T(\boldsymbol{y}_T - \boldsymbol{X}_T\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \boldsymbol{M}_T\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) + \lambda_1 s(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \\ \boldsymbol{0} & = & -\boldsymbol{M}_T^T(\boldsymbol{y}_T - \boldsymbol{X}_T\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) - \boldsymbol{M}_T\boldsymbol{\theta}(\boldsymbol{\lambda})) + \lambda_2\boldsymbol{D}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{array} \tag{25}$$

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 β_i. Let the active regression coefficients be $\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})$ and restricted design matrices be $\dot{\boldsymbol{X}}_T$ and $\dot{\boldsymbol{X}}_V$. The subgradient condition is therefore reduced to the following equation:

$$\boldsymbol{0} = -\dot{\boldsymbol{X}}_T^T(\boldsymbol{y}_T - \dot{\boldsymbol{X}}_T\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}) - \boldsymbol{M}_T\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) + \lambda_1 s(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \tag{26}$$

By the chain rule, the gradient is:

$$\nabla_\lambda L_V(\boldsymbol{\lambda})_i = -\left(\dot{\boldsymbol{X}}_V\frac{\partial\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})}{\partial\lambda_i} + \boldsymbol{M}_V\frac{\partial\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}{\partial\lambda_i}\right)^T\left(\boldsymbol{y}_V - \dot{\boldsymbol{X}}_V\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}) - \boldsymbol{M}_V\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\right) \text{ for } i = 1, 2 \tag{27}$$

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

$$\begin{array}{rcl} \frac{\partial}{\partial\lambda_1}\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}) & = & -\left(\dot{\boldsymbol{X}}_T^T\dot{\boldsymbol{X}}_T - \dot{\boldsymbol{X}}_T^T\boldsymbol{M}_T(\boldsymbol{M}_T^T\boldsymbol{M}_T + \lambda_2\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{M}_T^T\dot{\boldsymbol{X}}_T\right)^{-1} s(\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda})) \\ \frac{\partial}{\partial\lambda_1}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) & = & -\left(\boldsymbol{M}_T^T\boldsymbol{M}_T + \lambda_2\boldsymbol{D}^T\boldsymbol{D}\right)^{-1}\boldsymbol{M}_T^T\dot{\boldsymbol{X}}_T\frac{\partial}{\partial\lambda_1}\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial\lambda_2}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) & = & -\left(\boldsymbol{M}_T^T\boldsymbol{M}_T - \boldsymbol{M}_T^T\dot{\boldsymbol{X}}_T(\dot{\boldsymbol{X}}_T^T\dot{\boldsymbol{X}}_T)^{-1}\dot{\boldsymbol{X}}_T^T\boldsymbol{M}_T + \lambda_2\boldsymbol{D}^T\boldsymbol{D}\right)\boldsymbol{D}^T\boldsymbol{D}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \\ \frac{\partial}{\partial\lambda_2}\dot{\hat{\boldsymbol{\beta}}}(\boldsymbol{\lambda}) & = & -\left(\dot{\boldsymbol{X}}_T^T\dot{\boldsymbol{X}}_T\right)^{-1}\dot{\boldsymbol{X}}_T^T\boldsymbol{M}_T\frac{\partial}{\partial\lambda_2}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \end{array} \tag{28}$$

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 $corr(i,j) = 0.5^{|i-j|}$. The response vector y was constructed as

$$y = X\beta + \sigma\epsilon \tag{29}$$

where the first 15 values of the coefficient vector β are ones and the rest are zeroes and $\epsilon \sim N(0,1)$. σ 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 10×10, 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

$$y = \sum_{l=1}^{3} X^{(k)}\beta^{(k)} + \sigma\epsilon \tag{30}$$

where $\epsilon \sim N(0,1)$, $\beta^{(k)} = (1,2,3,4,5,0,...,0)$ for $k = 1,2,3$. σ 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 10×10 grid for grid-based cross validation, starting from 1e-5 to $\max(\{||X^{(k)T}y||_2\}_{l=1,...,m})$.

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.

$$\begin{aligned} x_i &= Z_1 + \epsilon_i \text{ for } i = 1,2,3 \\ x_i &= Z_2 + \epsilon_i \text{ for } i = 4,5,6 \\ x_i &\sim N(0,1) \text{ for } i = 7,...,20 \end{aligned} \tag{31}$$

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

$$y = X\beta + \kappa g(z) + \sigma\epsilon \tag{32}$$

The constants κ and σ were chosen such that the linear to nonlinear ratio $\frac{||X\beta||_2}{||g(Z)||_2}$ was 2 and the signal to noise ratio was 2, respectively. We set $\beta = (1,1,1,1,1,1,0,...,0)$ 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 10×10 grid for grid-based cross validation, with values ranging from $1e-6$ to 10.

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

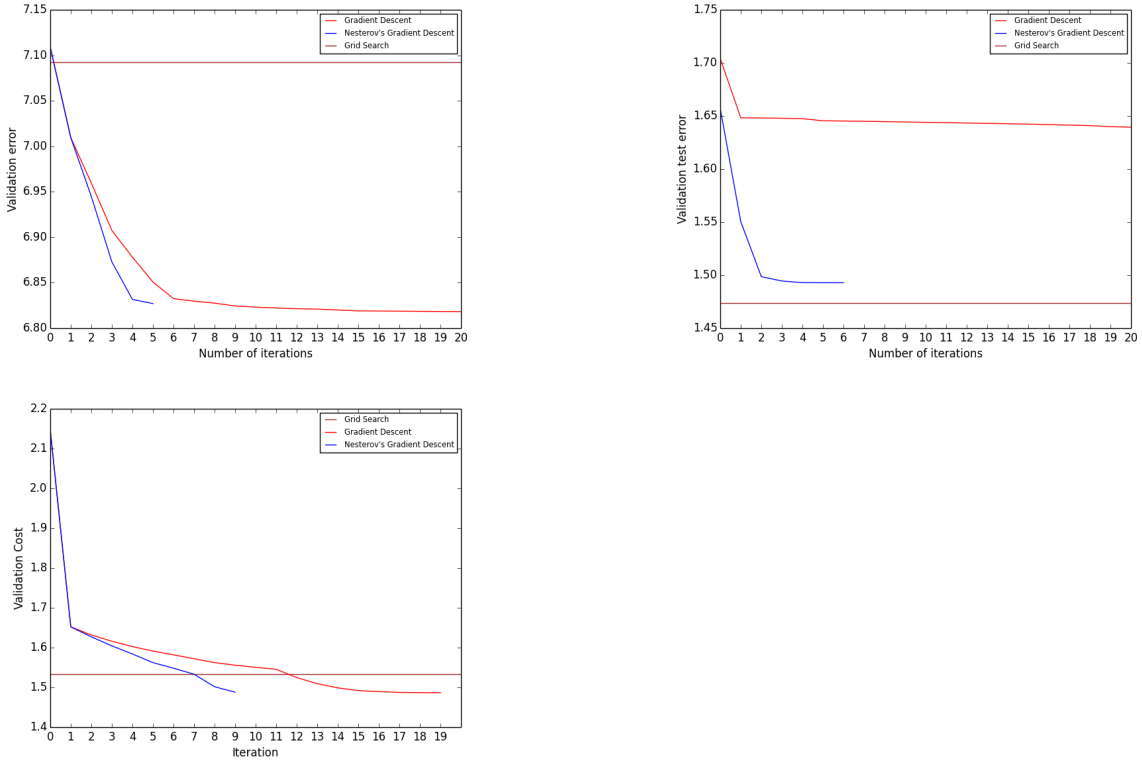

Figure 2. Validation error progression: Top Left - Elastic Net, Top Right - Grouped Lasso, Bottom Left - APLM

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.

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 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 compare the effectiveness of descent-based joint optimization on an "unpooled" variant of sparse group lasso to grid-based joint optimization on the regular sparse group lasso. Unpooled sparse group lasso increases the number of penalty parameters by splitting the group lasso penalty parameter λ_1 into a parameter for each covariate group:

$$\sum_{l=1}^{m} \lambda_1^{(k)} ||\beta^{(k)}||_2 + \lambda_2 ||\beta||_1 \tag{33}$$

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. This increased flexibility can potentially result in more accurate models when covariate groups have very different distributions. Since the number of regularization parameters are linear in the number of coefficient groups, we could test cases with over a hundred penalty parameters.

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. Descent-based joint optimization for unpooled sparse group lasso was initialized at 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, β error, which is defined as $||\beta - \hat{\beta}||_2^2$, and the percentage of nonzero coefficients correctly identified out of all the nonzero coefficients in the fitted model. As shown in Table 2, unpooled sparse group lasso performed better by all three metrics in all the experiments. Furthermore, the runtime for descent-based joint optimization for unpooled sparse group lasso is faster than grid-based joint optimization for two-parameter sparse group lasso, even when tuning 151 regularization parameters.

n=60, p=300, g=3, m=30				
	β Error	% correct nonzero β	Test Error	Runtime (sec)
Gridsearch (baseline)	1.13	10.70	0.04	15.81
Gradient Descent	0.18	23.79	0.01	5.62
n=60, p=1500, g=3, m=50				
	β Error	% correct nonzero β	Test Error	Runtime (sec)
Gridsearch (baseline)	7.79	9.63	0.28	148.64
Gradient Descent	4.00	17.79	0.14	88.78
n=60, p=1500, g=3, m=150				
	β Error	% correct nonzero β	Test Error	Runtime (sec)
Gridsearch (baseline)	2.20	10.69	0.080	162.14
Gradient Descent	0.06	15.34	0.002	48.63

Table 2. Unpooled sparse group lasso

4.2 Additive partial linear models with three regularization parameters

Most APLMs have at most two regularization parameters because of the computational intractability of having more. Joint optimization gives us the freedom to add many more penalty terms, so we tested whether adding a third penalty improves model performance. For this example, we replaced the lasso penalty with the elastic net penalty:

$$\lambda_1||\beta||_1 + \frac{1}{2}\lambda_2||\beta||_2^2 + \frac{1}{2}\lambda_3||D^{(2)}\theta||_2^2 \tag{34}$$

The experiment was run on simulated data generated using the settings previously specified. To adjust for the additional regularization parameter, joint optimization was initialized at the six points $\lambda_1 = \lambda_2 = \lambda_3 = 10^i$ for $i = -4, ..., 1$. We performed tests with three nonlinear functions of varying levels of smoothness.

Models were judged by their predictive accuracy and how closely they could estimate the linear and nonlinear components. For the latter, we used β error, defined as $||\beta - \hat{\beta}||_2^2$, and θ error, defined as $||g(z) - \theta||_2^2$.

The results in Table 3 show that using three penalty terms in the criterion significantly improved the models by all metrics. An improvement in predicting the linear regression coefficients β was expected since elastic net is well-suited for modeling sparse, correlated data. The improvement in the nonlinear model was less expected, since no additional regularization parameters were added for this. However, it seems that the additional regularization parameter for imposing structure to the linear component also properly imposed structure for the nonlinear component.

Joint optimization for three regularization parameters required a runtime slightly longer than grid search while, nonetheless, remaining very reasonable. The additional regularization parameter resulted in over a two-fold increase in the runtime for joint optimization. The increase in runtime can be attributed to the two additional initialization points and the additional cost of computing the gradient with respect to another regularization parameter. The most important thing to note is that this increase in runtime is not exponential in the number of parameters but, rather, polynomial (can we say this?). Grid search on the other hand would increase the runtime ten-fold for every additional parameter added.

Figure 3 compares example model fits from joint optimization and grid search. The plots of the linear regression coefficient vector show that fits from elastic net regularization tended to have similar nonzero coefficients for the highly-correlated covariates whereas fits from the lasso tended to set some of these coefficients to zero. In addition, when comparing the estimates for the nonlinear component, fits from the elastic net were smoother in general as compared to those from the lasso.

$g(z) = 4z^3 - z^2 + 2z$				
	β Error	θ error	Test Error	Runtime (sec)
Gridsearch	0.59	3.35	3.78	35.48
Gradient Descent	0.38	2.96	3.73	43.44
$g(z) = \sin(5z) + \sin(15(z-3))$				
	β Error	θ error	Test Error	Runtime (sec)
Gridsearch	0.51	3.76	3.90	37.04
Gradient Descent	0.34	3.73	3.79	45.95
$g(z) = (2-z)\sin(20z^4)$				
	β Error	θ error	Test Error	Runtime (sec)
Gridsearch	0.58	4.91	4.13	40.75
Gradient Descent	0.41	4.85	4.08	54.63

Table 3. Additive Partial Linear Model

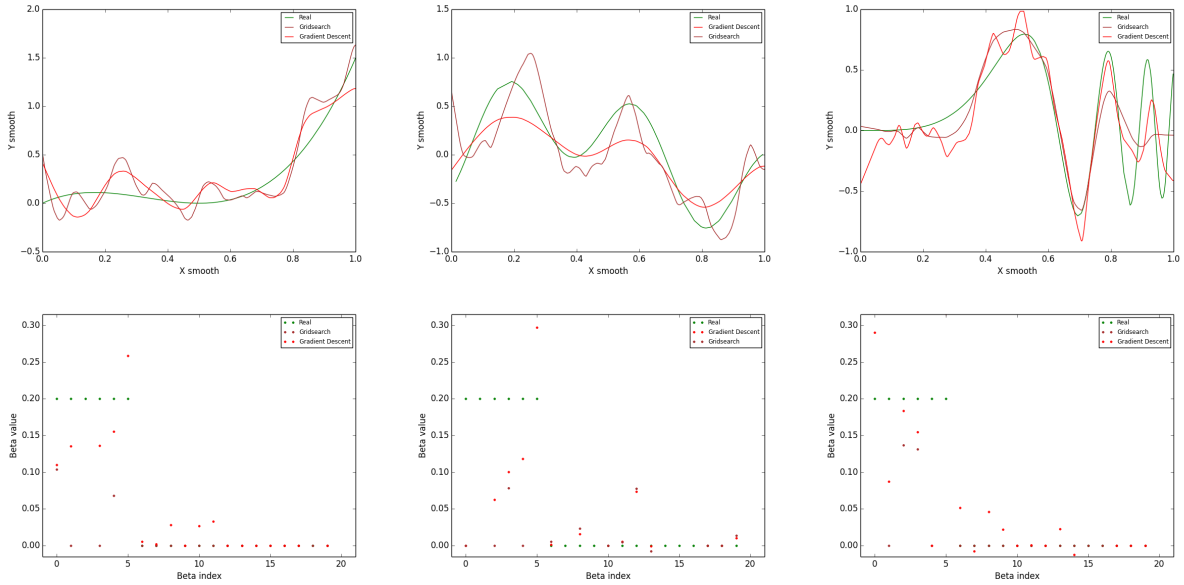

Figure 3. Top: Example fits to the three smooth functions used in the experiments. The green line is the true function. Bottom: Example fits for the linear regression coefficients. The green is the true beta.

	% 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

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.

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 5×5 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 criterions, but this method should be tested on more criterions 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

Lemma 1. *Let the function $\hat{f}(\cdot)$ be defined as:*

$$\hat{f}(\lambda) = \arg\min_{f} L(f, \lambda) \tag{35}$$

For a given λ_0, suppose there exists a neighborhood around $\hat{f}(\lambda_0)$ such that for every f in the neighborhood, $L(f, \lambda_0)$ is twice continuously differentiable with respect to f and the hessian $Hess_f(L(f, \lambda_0))|_{\hat{f}(\lambda_0)}$ is nonsingular. Then $\hat{f}(\cdot)$ is continuously differentiable in some neighborhood around λ_0.

Proof. Given that the loss function and penalty functions are differentiable, $\hat{f}(\lambda_0)$ is the point where the gradient of the function L is zero:

$$\nabla_f L(f, \lambda_0) = 0 \tag{36}$$

Let $H(f, \lambda_0)$ denote the expression on the left hand side. From our assumptions, we know that $H(f, \lambda_0)$ is continuously differentiable with respect to f in some neighborhood of $\hat{f}(\lambda_0)$ and the Jacobian matrix $D_f H(f, \lambda_0)|_{\hat{f}(\lambda_0)}$ is nonsingular. By the implicit function theorem, there exists a continuously differentiable function γ such that for every λ in some neighborhood around λ_0,

$$H(\gamma(\lambda), \lambda) = 0 \tag{37}$$

and $\hat{f}(\lambda) = \gamma(\lambda)$.

□

Lemma 2. *Let $\hat{f}(\cdot)$ be defined as in lemma 1. For a given λ_0, suppose there exists a neighborhood around $f(\lambda_0)$ such that the regularity conditions from lemma 1 for $L(f, \lambda_0)$ only hold with respect to the active set of f. Furthermore, suppose there exists a neighborhood around λ_0 such that the non-active set of $\hat{f}(\cdot)$ is locally constant. Then $\hat{f}(\cdot)$ is continuously differentiable in some neighborhood around λ_0.*

Proof. By the KKT conditions, $\hat{f}(\lambda_0)$ is the point where the following subgradient condition is satisfied:

$$0 \in \partial_f L(f, \lambda_0) \tag{38}$$

For the active set $\dot{f}$, the subgradient condition reduces to the gradient condition:

$$0 = \nabla_{\dot{f}} L(f, \lambda_0) \tag{39}$$

Applying Theorem 1 to the gradient condition in (39), we know that the active set $\dot{f}(\cdot)$ is continuously differentiable for some neighborhood around λ_0. By our assumptions, the partial derivative of $\hat{f}(\cdot)$ with respect to the non-active set is zero in some neighborhood around λ_0. Combining the active and non-active sets, we have that $\hat{f}(\cdot)$ is continuously differentiable in some neighborhood around λ_0.

□

6.2 Elastic Net Gradient Derivations

MATHS

6.3 Sparse Group Lasso Derivations

6.3.1 Unpooled Sparse Group Lasso Derivations

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

$$\begin{aligned}\nabla_{\lambda_1^{(k)}} L(\boldsymbol{y}_V, X_V\hat{\beta}) &= \frac{1}{n}\left(X_V(\frac{1}{n}X^TX + M_1 + M_2)^{-1} \begin{bmatrix} 0 \\ \vdots \\ \frac{\hat{\beta}^{(l}}{||\hat{\beta}^{(l}||_2} \\ \vdots \\ 0 \end{bmatrix} \right)^T (\boldsymbol{y}_V - X_V\hat{\beta}) \\ \nabla_{\lambda_2} L(\boldsymbol{y}_V, X_V\hat{\beta}) &= \frac{1}{n}\left((\frac{1}{n}X^TX + M_1 + M_2)^{-1} sign(\hat{\beta}) \right)^T (\boldsymbol{y}_V - X_V\hat{\beta})\end{aligned} \tag{40}$$

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

6.4 Additive Partial Linear Model Derivations

6.4.1 Additive Partial Linear Model Derivations: 3-penalties

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

$$\min_{\lambda_1,\lambda_2,\lambda_3} \frac{1}{2}\left|\left|\boldsymbol{y}_V - X_V\hat{\beta} - M_V\hat{\theta}\right|\right|_2^2$$
$$\text{where } \hat{\beta}, \hat{\theta} = \arg\min_{\beta,\theta} \frac{1}{2}\left|\left|y - X\beta - M_T\theta\right|\right|_2^2 + \lambda_1||\beta||_1 + \frac{1}{2}\lambda_2||\beta||_2^2 + \frac{1}{2}\lambda_3||D^{(2)}\theta||_2^2 \tag{41}$$

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

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

$$\nabla_{\lambda_i} L(\boldsymbol{y}_V, X_V\hat{\beta} + \hat{\theta}) = (X_V\frac{\partial\beta}{\partial\lambda_i} + M_V\frac{\partial\theta}{\partial\lambda_i})^T(\boldsymbol{y}_V - X_V\hat{\beta} - \hat{\theta}_V) \tag{42}$$

where

$$\begin{aligned}\frac{\partial\beta}{\partial\lambda_1} &= -(X_T^TX_T - X_T^TM_T(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T + \lambda_2I)^{-1}sign(\hat{\beta}) \\ \frac{\partial\theta}{\partial\lambda_1} &= -(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T\frac{\partial\beta}{\partial\lambda_1} \\ \frac{\partial\beta}{\partial\lambda_2} &= -(X_T^TX_T - X_T^TM_T(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T + \lambda_2I)^{-1}\hat{\beta} \\ \frac{\partial\theta}{\partial\lambda_2} &= -(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T\frac{\partial\beta}{\partial\lambda_2} \\ \frac{\partial\theta}{\partial\lambda_3} &= -(M_T^TM_T - M_T^TX_T(X_T^TX_T + \lambda_2I)^{-1}X_T^TM_T + \lambda_3D^TD)^{-1}D^TD\theta \\ \frac{\partial\beta}{\partial\lambda_3} &= -(X_T^TX_T + \lambda_2I)^{-1}X_T^TM_T\frac{\partial\theta}{\partial\lambda_3}\end{aligned} \tag{43}$$

Acknowledgments

BLAH

References

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