# 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 more efficiently by treating it as a continuous optimization problem and updating the parameter values using subgradient descent. We demonstrate that this method of "joint optimization" achieves the same performance as cross validation over a grid. Joint optimization enables us to test regularizations with many penalty parameters, through which we discover new regularization methods that attain superior accuracy. Our experiments are performed on simulated and real data.

## 1 Introduction

Consider the usual regression problem 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 many favorable 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:

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

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. We estimate this using cross-validation, which 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:

$$\begin{gathered} \min_{\boldsymbol{\lambda} \in \Lambda} L(y_V, \hat{f}(X_V | \boldsymbol{\lambda})) \\ \text{where } \hat{f}(\cdot | \boldsymbol{\lambda}) = \arg\min_f L(y_T, f(X_T)) + \sum_{i=1}^{k} \lambda_i J_i(f) \end{gathered} \tag{2}$$

The simplest approach to solving the problem above is brute force: models are fit over a grid of parameter values and the one with the lowest validation error is selected. As long as the grid is large and fine enough, this method of "grid search" will find a solution close to the global optimum. However, this approach 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 explicitly solving the continuous optimization problem in (2) over the penalty parameter space. We use subgradient descent to update the regularization parameters and use ordinary optimization techniques to solve for the model parameters at each iteration. Simulation studies show that this "joint optimization" of both model and penalty parameters 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 to run. 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. 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 joint 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 the joint optimization problem explicitly is still computationally tractable. Finally, we present results from experiments on real data in Section 5.

# 2 Descent-based Joint Optimization

## 2.1 Definition

Suppose that the data set has $n$ observations with $p$ predictors. 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 $(J_i, \lambda_i)$ is the $i$th penalty function and its corresponding penalty parameter.

As mentioned before, we use cross-validation to find the optimal penalty parameters $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_k)^T$. Let the data set be partitioned into training set $(\boldsymbol{y}_T, \boldsymbol{X}_T)$ and validation set $(\boldsymbol{y}_V, \boldsymbol{X}_V)$, respectively. The optimal penalty and model parameters are obtained by solving the following joint 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 sometimes use $L_V(\boldsymbol{\lambda})$ to denote $L(y_V, \hat{f}(X_V | \boldsymbol{\lambda}))$ since the validation loss is ultimately a function of the penalty parameters.

We propose solving this joint optimization problem explicitly using a descent-based method over the regularization parameter space. For example, in the simple case where the criterion is differentiable, the penalty parameters can be updated using gradient descent or some variant thereof. In fact, in many situations, even if the criterion is non-differentiable, we show that it is possible to update the penalty parameters using gradient descent.

In contrast, grid-based methods do not incorporate knowledge about the criterion function. Grid-based methods solve the joint optimization problem by fitting models over a $k$-dimensional grid $G$

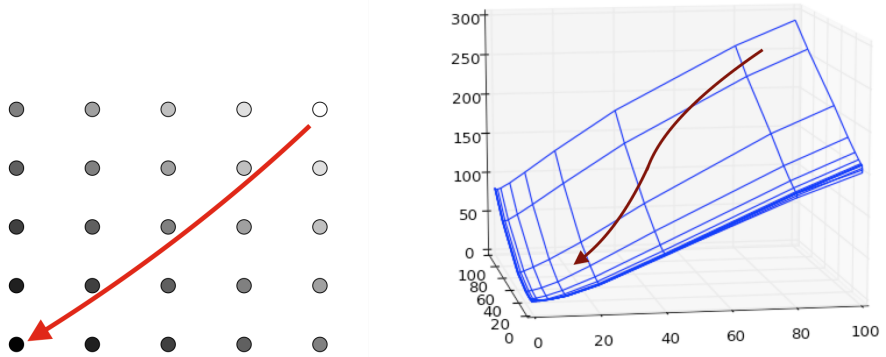


**Figure 1.** Left: darker points mean lower validation loss. Descent-based optimization descends in the most direct path towards the point producing the lowest validation loss. Right: The 3D version. We can tune regularization parameters using a grid search... or just descend opposite of the gradient.

in the penalty parameter space. Therefore, the computational runtime of grid-based methods grows exponentially with the number of parameters.

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 clearly 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 paper, we solve joint optimization problems using gradient descent. We first describe the procedure 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 $\alpha$.
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 $\nabla L_V(\boldsymbol{\lambda})$ since by the chain rule, the gradient requires calculating the gradient of the minimizer of penalized regression $\hat{f}$:

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

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

Usually there is no closed-form solution, so we rely on the KKT conditions for an implicit equation regarding the minimizer of the penalized regression. At optimality, the KKT conditions state that the gradient of the penalized criterion with respect to $\hat{f}$ is zero:

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

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.

We 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! Complicated!

Now we consider the case where $L_V(\boldsymbol{\lambda})$ and the penalty functions are non-differentiable. We show that if the loss function and penalty functions satisfy some basic assumptions, $L_V(\boldsymbol{\lambda})$ is differentiable almost everywhere and performing gradient descent is still sufficient in practice. The theorem is stated below:

**Theorem 1.** *BLAH*

Before presenting the proof, we note that the assumptions hold true in many cases. The theorem holds for many popular penalized regressions, such as ridge, elastic net, etc. BLAH BLAH BLAH. As the number of penalty functions and parameters increase, it is more likely for these assumptions to hold true (maybe?). The assumptions do not hold true for BLAH BLAH BLAH. However, in those cases, we can just use a grid-based method to find the optimal parameters.

*Proof.* Our goal is to show that $L_V(\lambda)$ is locally Lipschitz since by Rademacher's theorem, locally Lipschitz functions are differentiable almost everywhere. $L_V(\lambda)$ is the composite of $L_V(\hat{f}(\lambda))$. Since we know that $L_V$ is locally Lipschitz in $\hat{f}$, it suffices to show that $\hat{f}$ is locally Lipschitz with respect to $\lambda$. (The composite of locally Lipschitz functions is locally Lipschitz)

Instead of the KKT gradient condition, we must use the KKT sub-gradient condition:

$$0 \in \delta_{\hat{f}} \big( L(y_T, \hat{f}(X_T; \Lambda)) + \sum_{i=1}^{k} \lambda_i J_i(\hat{f}; \Lambda) \big) \tag{7}$$

We reduce this to the gradient KKT condition by restricting our equation set to just the active set of $\hat{f}$. That is, we solely deal with the active set, which is the model parameters $f_i$ where the derivative $\frac{\partial g}{\partial \hat{f}}$ exists. We can do this because BLAH. Therefore, we can reduced the KKT condition to its gradient form, as seen in Equation 6.

This implicit function is locally Lipschitz. By the implicit function theorem, there is a function $\hat{f}$ that is locally Lipschitz with respect to $\lambda$.

Done!

□

Therefore, since we have shown that at each iteration, we just need to restrict the problem to the active set and continue executing gradient descent.

### 2.2.3 Variants of Gradient Descent

Joint optimization problems can also be solved using variants of gradient descent. Instead of a constant step size $\alpha$, one can use decreasing or adaptive step sizes $\alpha(\lambda^{(k)})$. In our simulation studies, we implement 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 restart whenever the criterion increases in value. For more detail, refer to the detailed analysis by O'Donaghue and Candes.

## 2.3 Joint optimization for example regressions

In this subsection, we present examples of joint optimization problems and how to derive the gradients for updating the regularization parameters. We begin with the simple example of ridge regression, followed by elastic net, sparse group lasso, and additive partial linear models.

### 2.3.1 Ridge Regression

The joint optimization problem for ridge regression is:

$$\begin{gathered}\min_\lambda \tfrac{1}{2}||y_V - X_V\hat{\beta}(\lambda)||^2 \\ \text{where } \hat{\beta}(\lambda) = \arg\min_\beta \tfrac{1}{2}||y_T - X_T\beta||^2 + \lambda||\beta||_2^2\end{gathered} \tag{8}$$

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

$$\hat{\beta}(\lambda) = (X^TX + \lambda I)^{-1}X^Ty \tag{9}$$

Differentiating the above equation with respect to $\lambda$ gives us the gradient:

$$\begin{aligned}\nabla_\lambda L_V(\lambda) &= -(X_V\tfrac{\partial\hat{\beta}}{\partial\lambda})^T(y_V - X_V\hat{\beta}(\lambda)) \\ &= (X_V(X_T^TX_T + \lambda I)^{-1}\hat{\beta}(\lambda))^T(y_V - X_V\hat{\beta}(\lambda))\end{aligned} \tag{10}$$

### 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 $\lambda_1$ and $\lambda_2$ corresponding to each of these penalties:

$$\begin{gathered}\min_{\lambda_1,\lambda_2} \tfrac{1}{2}\left|\left|y_V - X_V\hat{\beta}(\lambda_1,\lambda_2)\right|\right|_2^2 \\ \text{where } \hat{\beta}(\lambda_1,\lambda_2) = \arg\min_\beta \tfrac{1}{2}\left|\left|y_T - X_T\beta\right|\right|_2^2 + \lambda_1||\beta||_1 + \tfrac{1}{2}\lambda_2||\beta||_2^2\end{gathered} \tag{11}$$

By the KKT conditions,

$$-X_T^T(y_T - X_T\hat{\beta}(\lambda_1,\lambda_2)) + \lambda_1 s(\hat{\beta}(\lambda_1,\lambda_2)) + \lambda_2\hat{\beta}(\lambda_1,\lambda_2) = 0 \tag{12}$$

where $s(\beta_i) = sign(\beta_i)$ if $\beta_i \neq 0$ and $s(\beta_i) = 0$ otherwise.

Performing implicit differentiation on the equation above gives us the subgradient:

$$\boldsymbol{g}_V(\lambda_1,\lambda_2) = \begin{bmatrix}(X_V^T(X_T^TX_T + \lambda_2 I)^{-1}s(\hat{\beta}(\lambda_1,\lambda_2)))^T(y_V - X_V\hat{\beta}(\lambda_1,\lambda_2)) \\ (X_V^T(X_T^TX_T + \lambda_2 I)^{-1}\hat{\beta}(\lambda_1,\lambda_2))^T(y_V - X_V\hat{\beta}(\lambda_1,\lambda_2))\end{bmatrix} \tag{13}$$

### 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. Let $m$ be the number of covariate groups, and split the predictors $X$ into submatrices $X^{(l)}$ and coefficient vector $\beta$ into vectors $\beta^{(l)}$ to correspond with each covariate group $l = 1, ..., m$. Sparse group lasso enforces group-level sparsity and within-group sparsity using the penalties $\sum_{l=1}^{m} ||\beta^{(l)}||_2$ and $||\beta||_1$. Joint optimization seeks the optimal regularization parameters $\lambda_1$ and $\lambda_2$:

$$\begin{gathered} \min_{\lambda_1,\lambda_2} \tfrac{1}{2n} \Big|\Big|y_V - \sum_{l=1}^{m} X_V^{(l)} \hat{\beta}^{(l)}(\lambda_1, \lambda_2)\Big|\Big|_2^2 \\ \text{where } \hat{\beta}(\lambda_1, \lambda_2) = \arg\min_{\beta} \tfrac{1}{2n} \Big|\Big|y_T - \sum_{l=1}^{m} X_T^{(l)} \beta^{(l)}\Big|\Big|_2^2 + \lambda_1 \textstyle\sum_{l=1}^{m} ||\beta^{(l)}||_2 + \lambda_2 ||\beta||_1 \end{gathered} \tag{14}$$

The KKT conditions state that

$$-X_T^T(y_T - X_T\hat{\beta}(\lambda_1, \lambda_2)) + \lambda_1 \sum_{l=1}^{m} \frac{\hat{\beta}(\lambda_1, \lambda_2)^{(l)}}{||\hat{\beta}(\lambda_1, \lambda_2)^{(l)}||_2} + \lambda_2 s(\hat{\beta}(\lambda_1, \lambda_2)) = 0 \tag{15}$$

Differentiating the KKT conditions gives the subgradient:

$$\boldsymbol{g}_V(\lambda_1, \lambda_2) = \begin{bmatrix} \frac{1}{n}\left(X_V(\frac{1}{n}X^TX - \lambda_1 M_1 + \lambda_1 M_2)^{-1} \begin{bmatrix} \frac{\hat{\beta}^{(1)}}{||\hat{\beta}^{(1)}||_2} \\ \vdots \\ \frac{\hat{\beta}^{(m}}{||\hat{\beta}^{(m}||_2} \end{bmatrix}\right)^T (y_V - X_V\hat{\beta}) \\ \frac{1}{n}\left(X_V(\frac{1}{n}X^TX - \lambda_1 M_1 + \lambda_1 M_2)^{-1} s(\hat{\beta})\right)^T (y_V - X_V\hat{\beta}) \end{bmatrix} \tag{16}$$

where $M_1$ is the block diagonal matrix with components $||\hat{\beta}^{(l)}||_2^{-3} \begin{bmatrix} \hat{\beta}_1^{(l)} & 0 & \\ 0 & \hat{\beta}_2^{(l)} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} — & \hat{\beta}^{(l)T} & — \\ — & \hat{\beta}^{(l)T} & — \\ \vdots & & \end{bmatrix}$

and $M_2$ is the block diagonal matrix with components $||\hat{\beta}^{(l)}||_2^{-1} I$ for $l = 1, ..., m$ from top left to bottom right.

(Using the notation $\hat{\beta}(\lambda_1, \lambda_2)$) is very hard to read...)

### 2.3.4 Additive Partial Linear Models

Additive partial linear models (APLMs), a semi-parametric regression model, is useful when the response is believed to depend on some of the predictors in a linear manner and the rest in a nonlinear manner. Given predictors $X \in \mathbb{R}^{n\times p}$ and $z \in \mathbb{R}^{n\times 1}$, APLMs predict the response $y$ as $x^T\beta + g(z)$, where $\beta$ is the vector of regression coefficients and $g : \mathbb{R} \to \mathbb{R}$ is some nonlinear function.

For this example, we specifically consider the case when the criterion contains a lasso penalty for the linear component and an $\ell_2$ trend filtering penalty for the nonlinear component. The joint optimization problem is as follows:

$$\begin{gathered} \min_{\lambda_1,\lambda_2} \tfrac{1}{2} \Big|\Big|y_V - X_V\hat{\beta} - M_V\hat{\theta}\Big|\Big|_2^2 \\ \text{where } \hat{\beta}, \hat{\theta} = \arg\min_{\beta,\theta} \tfrac{1}{2} \Big|\Big|y - X\beta - M_T\theta\Big|\Big|_2^2 + \lambda_1 ||\beta||_1 + \tfrac{1}{2}\lambda_2 ||D\theta||_2^2 \end{gathered} \tag{17}$$

where $M_V$ is the matrix such that $M_V X = X_V$ and $M_T$ is the matrix such that $M_T X = X_T$ .

$D$ is the second-order difference matrix for unevenly spaced inputs $z$ (Cite Ryan's paper?):

$$D = D^{(1)} \cdot diag\big(\frac{1}{z_2 - z_1}, \frac{1}{z_3 - z_2}, ..., \frac{1}{z_n - z_{n-1}}, 0\big) \cdot D^{(1)} \tag{18}$$

where

$$D^{(1)} = \begin{bmatrix} -1 & 1 & 0 & ... & 0 & 0 \\ 0 & -1 & 1 & ... & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & ... & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{19}$$

The KKT conditions state that:

$$\begin{array}{rcl} -X_T^T(y_T - X_T\beta - M_T\theta) + \lambda_1 s(\beta) & = & 0 \\ -M_T^T(y_T - X_T\beta - M_T\theta) + \lambda_2 D\theta & = & 0 \end{array} \tag{20}$$

By the chain rule, we know that the components of the subgradient are:

$$\boldsymbol{g}_V(\lambda_1, \lambda_2, \lambda_3)_i = -(X_V \frac{\partial \beta}{\partial \lambda_i} + M_V \frac{\partial \theta}{\partial \lambda_i})^T (y_V - X_V\hat{\beta} - \hat{\theta}_V) \tag{21}$$

and the partial (sub)derivatives (abuse of notation much?), from implicit subdifferentiation of the KKT conditions, are:

$$\begin{array}{rcl} \frac{\partial \beta}{\partial \lambda_1} & = & -(X_T^T X_T - X_T^T M_T (M_T^T M_T + \lambda_2 D^T D)^{-1} M_T^T X_T)^{-1} s(\hat{\beta}) \\ \frac{\partial \theta}{\partial \lambda_1} & = & -(M_T^T M_T + \lambda_2 D^T D)^{-1} M_T^T X_T \frac{\partial \beta}{\partial \lambda_1} \\ \frac{\partial \theta}{\partial \lambda_2} & = & -(M_T^T M_T - M_T^T X_T (X_T^T X_T)^{-1} X_T^T M_T + \lambda_2 D^T D) D^T D\theta \\ \frac{\partial \beta}{\partial \lambda_2} & = & -(X_T^T X_T)^{-1} X_T^T M_T \frac{\partial \theta}{\partial \lambda_2} \end{array} \tag{22}$$

# 3 Results: validation error minimization

We first address the question of whether or not joint optimization can achieve validation loss values as low as grid search. The key metrics in this comparison are, obviously, the average validation loss values for the fitted models and runtime.

We ran our experiments for the example regressions elastic net, sparse group lasso, and APLMs on simulated data. Joint optimization was implemented using gradient descent and Nesterov's gradient descent with adaptive restarts to understand how different implementations affects results. We first delineate the simulation settings and follow with a discussion of the results.

## 3.1 Elastic net

We generated 30 datasets with 80 training and 20 validation observations each and 250 predictors. For the coefficient vector $\beta$, the first 15 values were ones and the rest were zeroes. The pairwise correlation between $x_i$ and $x_j$ was set to be $corr(i, j) = 0.5^{|i-j|}$. The response vector $y$ was constructed as

$$y = X\beta + \sigma\epsilon \tag{23}$$

where $\epsilon \sim N(0, 1)$. $\sigma$ was chosen such that the signal to noise ratio is 2.

Joint optimization was initialized at two points (0.01, 0.01) and (10, 10). Grid search was implemented over a $10 \times 10$ grid, 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

$$y = \sum_{l=1}^{3} X^{(l)}\beta^{(l)} + \sigma\epsilon \tag{24}$$

| 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

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

Joint optimization was initialized at three points, where all regularization parameters were set to 0.01, 1, and 100. Grid search used a $10 \times 10$ grid, starting from 1e-5 to $\max(\{||X^{(l)T}y||_2\}_{l=1,...,m})$.

### 3.3 Additive partial linear models

We generated 30 datasets with 100 training and 25 validation observations and 21 predictors. The first 20 predictors was the input for 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.

$$\begin{aligned} x_i &= Z_1 + \epsilon_i \text{ for } i = 1,2,3 \\ x_i &= Z_2 + \epsilon_i \text{ for } i = 4,5,6 \\ x_i &\sim N(0,1) \text{ for } i = 7,...,20 \end{aligned} \tag{25}$$

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

$$y = X\beta + \kappa g(z) + \sigma\epsilon \tag{26}$$

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 $\beta = (1,1,1,1,1,1,0,...,0)$ and $g(z) = (2-z)\sin(20z^4)$.

Joint optimization was initialized at the four points $\lambda_1 = \lambda_2 = 10^i$ for $i = -2,...,1$. Grid search was performed over a $10 \times 10$ grid, with values ranging from $1e-6$ to 10.

### 3.4 Discussion of results

Joint optimization and grid search had the same performance in the experiments for all three example regressions. As shown in Table 1 , the only difference between grid search and joint optimization is the runtime. As hypothesized, subgradient descent and Nesterov's subgradient descent were much faster than grid search. For all the regression examples, subgradient descent was over 40% faster and Nesterov's subgradient descent was over 50% faster.

In addition, Figure 2 are example plots of how validation error decreases as joint optimization tunes the regularization parameters. The lowest validation error achieved by grid search is plotted as a

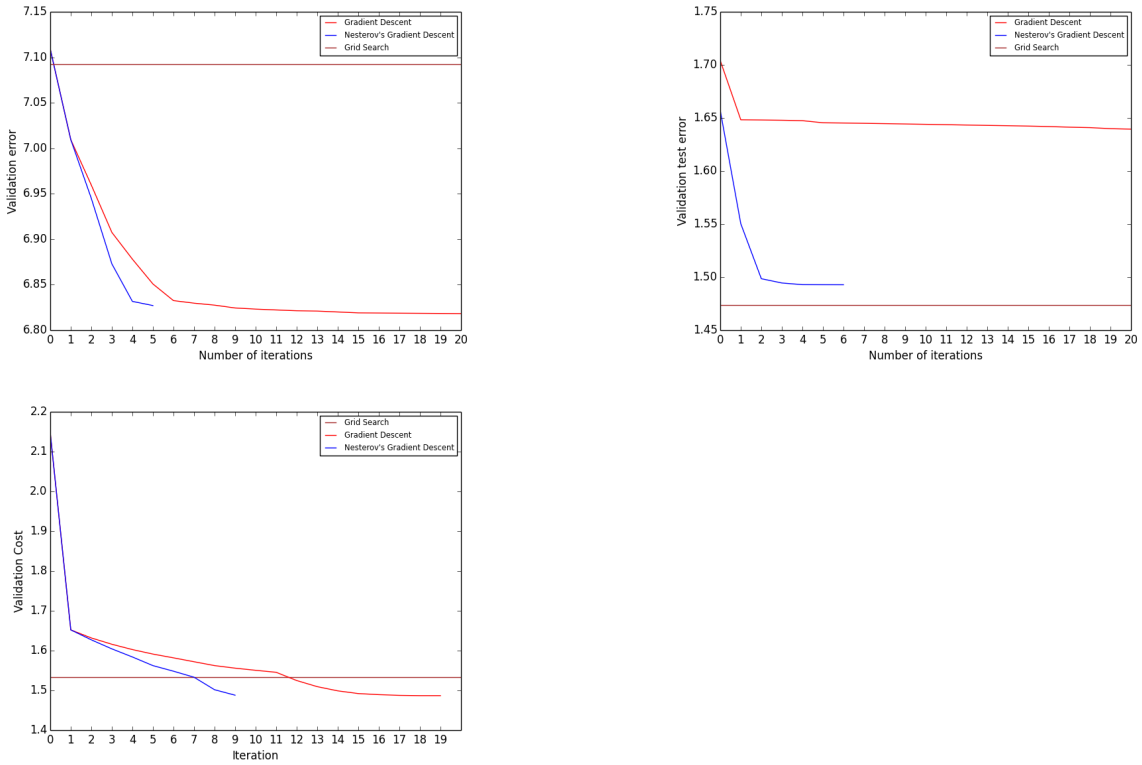


**Figure 2.** Validation error progression: Top Left - Elastic Net, Top Right - Grouped Lasso, Bottom Left - APLM

horizontal line as reference. In general, Nesterov's subgradient descent requires fewer iterations to find the local minima.

(Do we need to be concerned that we have only shown that grid search and joint optimization produces similar results for criterions with 2 regularization parameters)

# 4 Results: regularizations with more than two penalties

This section shows that joint optimization can effectively and efficiently solve regressions with over two regularization parameters.

We created more complex criterions for the experiments by generalizing the two of the previously mentioned regression examples. For the sparse group lasso, we increased the number of regularization parameters to one plus the number of covariate groups. For the APLM example, we increased the number of regularization parameters to three. For each example, we fitted models using joint optimization and compared against models produced by grid search applied to the original two-penalty problem.

Joint optimization formulations and gradient derivations of these criterions with are included in the Appendix.

## 4.1 Unpooled sparse group lasso

In sparse group lasso, because all $m$ covariate groups share the same two regularization parameters, the thresholds for setting coefficients and coefficient groups to zero are also shared (CITE). Here, we propose a variant of sparse group lasso such that each covariate group has an individual regularization parameter. This generalization allows for different thresholds for each coefficient and coefficient group, which is a more accurate assumption when modeling covariate groups with very different distributions. The new penalty, which we call "unpooled" sparse group lasso, replaces the original $\lambda_1$ parameter with a set of parameters $\{\lambda_1^{(l)}\}_{l=1,\ldots,m}$:

$$\sum_{l=1}^{m} \lambda_1^{(l)} ||\beta^{(l)}||_2 + \lambda_2 ||\beta||_1 \tag{27}$$

To compare the performance of regularization to the original sparse group lasso, we simulate data using the same settings as before. The regularization parameters for unpooled sparse group lasso were 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. In total, we performed three experiments, varying the number of covariates and the number of covariate groups $m$.

We measured three metrics for model performance: test error, $\beta$ error, which is defined as $||\beta - \hat{\beta}||_2^2$, and the number of nonzero coefficients correctly identified out of the total number of nonzero coefficients in the fitted model.

As seen in Table 2, unpooled sparse group lasso performs better by all metrics in all the experiments. The runtime for joint optimization, even when there are 150 regularization parameters, is still faster than running grid search over the two regularization parameters in 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:

$$\lambda_1 ||\beta||_1 + \frac{1}{2}\lambda_2 ||\beta||_2^2 + \frac{1}{2}\lambda_3 ||D^{(2)}\theta||_2^2 \tag{28}$$

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

| 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

$\lambda_1 = \lambda_2 = \lambda_3 = 10^i$ 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 $\beta$ error, defined as $||\beta - \hat{\beta}||_2^2$, and $\theta$ 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 $\beta$ 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.

# 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

| $g(z) = 4z^3 - z^2 + 2z$ | | | | |
|---|---|---|---|---|
| | $\beta$ 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

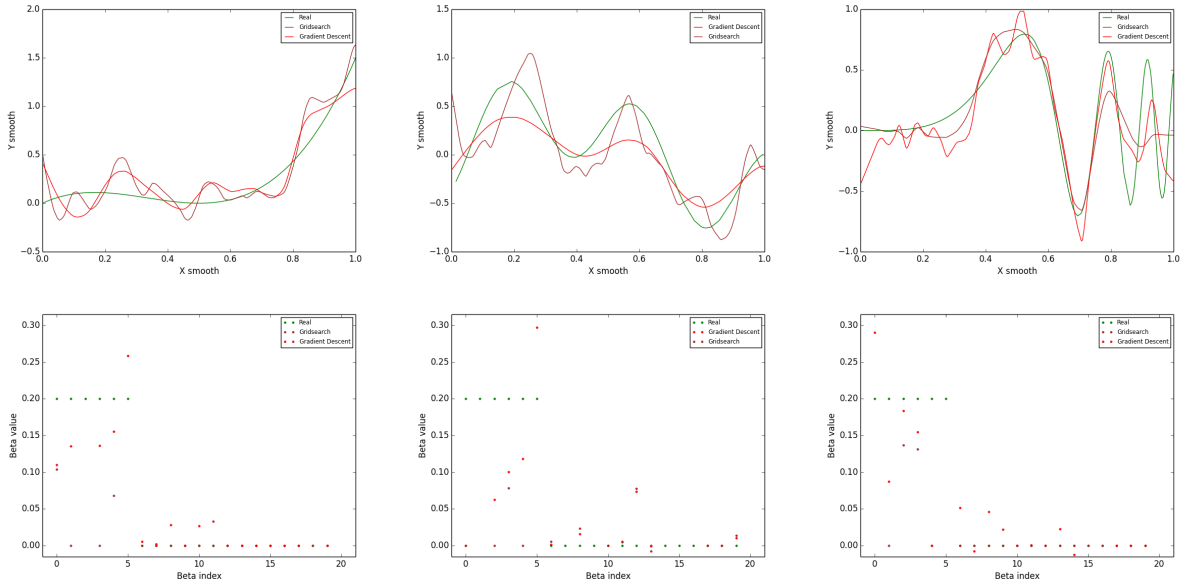


**Figure 3.** Top: Example fits to the three smooth functions used in the experiments. The green line is the true function. Bottom: Example fits for the linear regression coefficients. The green is the true beta.

| | % 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

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 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 Elastic Net Gradient Derivations

MATHS

## 6.2 Sparse Group Lasso Derivations

### 6.2.1 Unpooled Sparse Group Lasso Derivations

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

$$\begin{aligned}\nabla_{\lambda_1^{(l)}} L(y_V, X_V\hat{\beta}) &= \frac{1}{n}\left( X_V(\frac{1}{n}X^TX + M_1 + M_2)^{-1} \begin{bmatrix} 0 \\ \vdots \\ \frac{\hat{\beta}^{(l}}{||\hat{\beta}^{(l}||_2} \\ \vdots \\ 0 \end{bmatrix} \right)^T (y_V - X_V\hat{\beta}) \\ \nabla_{\lambda_2} L(y_V, X_V\hat{\beta}) &= \frac{1}{n}\left( (\frac{1}{n}X^TX + M_1 + M_2)^{-1} sign(\hat{\beta}) \right)^T (y_V - X_V\hat{\beta})\end{aligned} \tag{29}$$

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

## 6.3 Additive Partial Linear Model Derivations

### 6.3.1 Additive Partial Linear Model Derivations: 3-penalties

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

$$\min_{\lambda_1,\lambda_2,\lambda_3} \frac{1}{2}\big|\big|y_V - X_V\hat{\beta} - M_V\hat{\theta}\big|\big|_2^2$$

$$\text{where } \hat{\beta}, \hat{\theta} = \arg\min_{\beta,\theta} \frac{1}{2}\big|\big|y - X\beta - M_T\theta\big|\big|_2^2 + \lambda_1||\beta||_1 + \frac{1}{2}\lambda_2||\beta||_2^2 + \frac{1}{2}\lambda_3||D^{(2)}\theta||_2^2 \tag{30}$$

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:

$$\nabla_{\lambda_i} L(y_V, X_V\hat{\beta} + \hat{\theta}) = (X_V\frac{\partial\beta}{\partial\lambda_i} + M_V\frac{\partial\theta}{\partial\lambda_i})^T(y_V - X_V\hat{\beta} - \hat{\theta}_V) \tag{31}$$

where

$$\begin{aligned}\frac{\partial\beta}{\partial\lambda_1} &= -(X_T^TX_T - X_T^TM_T(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T + \lambda_2I)^{-1}sign(\hat{\beta}) \\ \frac{\partial\theta}{\partial\lambda_1} &= -(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T\frac{\partial\beta}{\partial\lambda_1} \\ \frac{\partial\beta}{\partial\lambda_2} &= -(X_T^TX_T - X_T^TM_T(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T + \lambda_2I)^{-1}\hat{\beta} \\ \frac{\partial\theta}{\partial\lambda_2} &= -(M_T^TM_T + \lambda_3D^TD)^{-1}M_T^TX_T\frac{\partial\beta}{\partial\lambda_2} \\ \frac{\partial\theta}{\partial\lambda_3} &= -(M_T^TM_T - M_T^TX_T(X_T^TX_T + \lambda_2I)^{-1}X_T^TM_T + \lambda_3D^TD)^{-1}D^TD\theta \\ \frac{\partial\beta}{\partial\lambda_3} &= -(X_T^TX_T + \lambda_2I)^{-1}X_T^TM_T\frac{\partial\theta}{\partial\lambda_3}\end{aligned} \tag{32}$$

## Acknowledgments

BLAH

## References

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