Gradient-based Regularization Parameter Selection for Problems with Non-smooth Penalty Functions

Jean Feng[*]

Department of Biostatistics, University of Washington

and

Noah Simon

Department of Biostatistics, University of Washington

January 23, 2017

Abstract

In high-dimensional and/or non-parametric regression problems, regularization (or penalization) is used to control model complexity and induce desired structure. Each penalty has a weight parameter that indicates how strongly the structure corresponding to that penalty should be enforced. Typically the parameters are chosen to minimize the error on a separate validation set using a simple grid search or a gradient-free optimization method. It is more efficient to tune parameters if the gradient can be determined, but this is often difficult for problems with non-smooth penalty functions. Here we show that for many penalized regression problems, the validation loss is actually smooth almost-everywhere with respect to the penalty parameters. We can therefore apply a modified gradient descent algorithm to tune parameters. Through simulation studies on example regression problems, we find that increasing the number of penalty parameters and tuning them using our method can decrease the generalization error.

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

1 Introduction

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

In recent years, there has been much interest in combining regularization methods to produce models with multiple desired characteristics. For example, the elastic net (Zou & Hastie 2003) combines the lasso and ridge penalties and the sparse group lasso (Simon et al. 2013) combines the group lasso and lasso penalties. In Bayesian regression, a popular method for pruning irrelevant features is to use automatic relevance determination, which associates each feature with a separate regularization parameter (Neal 1996). Finally, neural networks commonly use regularization to control the weights at each node. Snoek et al. (2012) showed that using separate regularization parameters for each layer in a neural network can actually improve performance. From a theoretical viewpoint, certain regression problems require multiple regularization parameters to achieve oracle convergence rates, such as additive models with different smoothness (van de Geer & Muro 2014). The general form of these regression problems is:

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

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

Regularization parameters control the degree of various facets of model complexity, such as the amount of sparsity or smoothness. Often the goal is to set the parameters to minimize the fitted model's generalization error. One usually estimates this using a training/validation

approach (or cross validation). In this approach, one fits a model on a training set $(\boldsymbol{X}_T, \boldsymbol{y}_T)$ and measures the model's error on a validation set $(\boldsymbol{X}_V, \boldsymbol{y}_V)$. The goal then is to choose penalty parameters $\boldsymbol{\lambda}$ that minimize the validation error, as formulated in the following joint optimization problem:

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

Here Λ is some set that $\boldsymbol{\lambda}$ are known to be in, which is often just $\mathbb{R}_+^J$. We will refer to finding $\hat{f}(\cdot | \boldsymbol{\lambda})$ as solving the inner optimization problem.

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

More efficient methods treat (2) as a continuous optimization problem, usually through a gradient-free or gradient-based approach. Gradient-free approaches include the Nelder-Mead simplex algorithm (Nelder & Mead 1965) and Bayesian optimization (Snoek et al. 2012, Bergstra et al. 2011, Hutter et al. 2011). Although Bayesian optimization is currently the gold standard in machine learning, gradient-free methods are generally unable to tune more than twenty or so parameters whereas gradient-based methods can handle hundreds or even thousands of parameters. To calculate the gradient, one can use reverse-mode differentiation through the entire training procedure (Maclaurin et al. 2015) or implicit differentiation of the KKT conditions (Larsen et al. 1998, Bengio 2000, Foo et al. 2008, Lorbert & Ramadge 2010). Existing implicit differentiation methods all require the optimization criterion to be smooth. In this paper we show that many problems for which the inner optimization problem is non-smooth can be reformulated in a way that makes them amenable to tuning parameter optimization via gradient descent.

In Section 2, we show that for certain joint optimization problems with non-smooth penalties, the outer optimization problem is still smooth almost everywhere. By locally reformulating the problem, we can apply the same implicit differentiation trick to obtain the gradient of the validation loss with respect to the penalty parameters. A descent-based

3

algorithm is then proposed for tuning the penalty parameters. Section 3 presents simulation studies comparing our method to gradient-free methods on regression problems with two to a hundred penalty parameters. We find that our method has lower generalization error compared to models trained with only two penalty parameters and outperforms gradient-free approaches. Section 4 applies our method to gene expression data to predict colitis status. The resulting models are significantly more sparse without sacrificing classification accuracy.

2   Gradient-based Joint Optimization

2.1   Definition

In this manuscript we will restrict ourselves to classes $\mathcal{F} = \{f_{\boldsymbol{\theta}} | \boldsymbol{\theta} \in \Theta\}$, which, for a fixed sample size n, are in some finite dimensional space Θ. This is not a large restriction: the class of linear functions meets this requirement; as does any class of finite dimensional parametric functions. Even non-parametric methods generally either use a growing basis expansion (e.g. Polynomial regression, smoothing-splines, wavelet-based-regression, locally-adaptive regression splines (Tsybakov 2008, Wahba 1981, Donoho & Johnstone 1994, Mammen et al. 1997)), or only evaluate the function at the observed data-points (eg. trend filtering, fused lasso, (Kim et al. 2009, Tibshirani et al. 2005)). In these non-parametric problems, for any fixed n, $\mathcal{F}$ is representable as a finite dimensional class. We can therefore rewrite (1) in the following form:

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

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

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

Note that joint optimization for K-fold cross validation is very similar. The outer criterion is the average validation loss over the models trained from all K folds. See the Appendix for

the full formulation.

For the remainder of the manuscript we will assume that the training criterion (3) is convex and has a unique minimizer. Also, we will assume that $L\left(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V)\right)$ is differentiable in $\boldsymbol{\theta}$. This assumption is met if both 1) $f_{\boldsymbol{\theta}}(\boldsymbol{X}_V)$ is continuous as a function of $\boldsymbol{\theta}$; and 2) $L\left(\boldsymbol{y}_V, \cdot\right)$ is smooth. Examples include the squared-error, logistic, and poisson loss functions, though not the hinge loss.

2.2 Smooth Training Criterion

Here we present a brief summary of the approach of applying gradient descent when the training criterion is smooth. For more details, refer to Bengio (2000).

Let the training criterion be denoted as

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

To calculate the gradient, we apply the chain rule

$$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right) = \left[\left.\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V))\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}\right]^{\top} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \tag{6}$$

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

$$\nabla_{\theta} L_T(\boldsymbol{\theta}, \boldsymbol{\lambda})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})} = \mathbf{0}. \tag{7}$$

Taking the derivative of both sides of (7) in $\boldsymbol{\lambda}$ and solving for $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$, we get:

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = -\left.\left[\nabla_{\theta}^2 L_T(\boldsymbol{\theta}, \boldsymbol{\lambda})^{-1} \nabla_{\boldsymbol{\theta}} P(\boldsymbol{\theta})\right]\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})} \tag{8}$$

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

We can plug (8) into (6) to get $\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)$. Note that because $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is defined in terms of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$, each gradient step requires minimizing the training criterion first. The gradient descent algorithm to solve (4) is given in Algorithm 1.

Algorithm 1 Gradient Descent for Smooth Training Criteria

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

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

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

Calculate the derivative of the model parameters with respect to the regularization parameters

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

Calculate the gradient

$$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)\Big|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(k)}} = \left[\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V))\Big|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}^{(k)})} \right]^\top \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(k)}} \tag{10}$$

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

$$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} - t^{(k)} \nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right)\Big|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(k)}} \tag{11}$$

end for

2.3 Nonsmooth Training Criterion

When the penalized training criterion in the joint optimization problem is not smooth, gradient descent cannot be directly applied. Nonetheless, we find that in many problems, the solution $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is smooth at almost every $\boldsymbol{\lambda}$ (eg. Lasso, Group Lasso, Trend Filtering); this means that we can indeed apply gradient descent in practice. In this section, we characterize these problems that are almost everywhere smooth. In addition, we provide a solution for deriving $\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ since calculating the gradient is a challenge in and of itself. This is then incorporated into an algorithm for tuning $\boldsymbol{\lambda}$ using gradient descent.

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

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

$$\Omega^L(\boldsymbol{\eta}) = \left\{ \boldsymbol{u} \middle| \lim_{\epsilon \to 0} \frac{L(\boldsymbol{\eta} + \epsilon \boldsymbol{u}) - L(\boldsymbol{\eta})}{\epsilon} \ \text{exists} \right\} \tag{12}$$

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

$$\underset{\boldsymbol{\theta} \in \Theta}{\arg \min}\, L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \underset{\boldsymbol{\theta} \in S}{\arg \min}\, L(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{13}$$

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

$$_{\boldsymbol{B}}\nabla f = \begin{pmatrix} \frac{\partial f}{\partial \boldsymbol{b}_1} \\ \frac{\partial f}{\partial \boldsymbol{b}_2} \\ \vdots \\ \frac{\partial f}{\partial \boldsymbol{b}_q} \end{pmatrix} \in \mathbb{R}^p; \quad _{\boldsymbol{B}}\nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial b_1^2} & \frac{\partial^2 f}{\partial b_1 \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_1 \partial b_q} \\ \frac{\partial^2 f}{\partial b_2 \partial b_1} & \frac{\partial^2 f}{\partial b_2^2} & \cdots & \frac{\partial^2 f}{\partial b_2 \partial b_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial b_q \partial b_1} & \frac{\partial^2 f}{\partial b_q \partial b_2} & \cdots & \frac{\partial^2 f}{\partial b_q^2} \end{pmatrix} \in \mathbb{R}^{p \times p} \tag{14}$$

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

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

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

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

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

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

Theorem 1 *Suppose our optimization problem is of the form in (4), with $L_T(\boldsymbol{\theta},\boldsymbol{\lambda})$ defined as in (5).*

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

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

$$\nabla_{\boldsymbol{\lambda}} L\left(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)\right) = \left[\left.\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{y}_V, f_{\boldsymbol{\theta}}(\boldsymbol{X}_V))\right|_{\boldsymbol{\theta}=\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda})}\right]^{\top} \frac{\partial}{\partial \boldsymbol{\lambda}} \tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \tag{15}$$

where

$$\tilde{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))} L_T(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{16}$$

We can therefore construct a gradient descent procedure based on the model parameter constraint in (16). At each iteration, let matrix $\boldsymbol{U}$ have orthonormal columns spanning the differentiable space $\Omega^{L_T(\cdot,\boldsymbol{\lambda})}(\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$. Since this space is also a local optimality space, it is sufficient to minimize the training criterion over the column space of $\boldsymbol{U}$. The joint optimization problem can be reformulated using $\boldsymbol{U}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ as the model parameters instead:

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \Lambda} L(\boldsymbol{y}_V, f_{\boldsymbol{U}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) \\ &\text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \operatorname*{arg\,min}_{\boldsymbol{\beta}} L_T(\boldsymbol{U}\boldsymbol{\beta}, \boldsymbol{\lambda}) \end{aligned} \tag{17}$$

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

$$\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \boldsymbol{U}\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \tag{18}$$

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

2.4 Examples

To better understand the proposed gradient descent procedure, we present example joint optimization problems with nonsmooth criteria and their corresponding gradient calculations.

For ease of notation, we will let $S_{\boldsymbol{\lambda}}$ denote the differentiable space of $L_T(\cdot, \boldsymbol{\lambda})$ at $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$. All the example regressions satisfy the conditions in Theorem 1; details are included in the Appendix. Note that in some examples below, we add a ridge penalty with a fixed small coefficient $\epsilon > 0$ to ensure that the problem satisfies Condition 3. Intuitively, the additional ridge penalty makes the models more well-behaved. For example, in the elastic net, combining the ridge and lasso penalties results in models that often exhibit better properties than just the lasso alone.

2.4.1 Elastic Net

The elastic net (Zou & Hastie 2003) is a linear combination of the lasso and ridge penalties that encourages both sparsity and grouping of predictors. We tune the regularization parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^{\top}$ according to the joint optimization problem:

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}_+^2} \tfrac{1}{2}\|\boldsymbol{y}_V - \boldsymbol{X}_V\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\|^2$$
$$\text{s.t. } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_T\boldsymbol{\theta}\|^2 + \lambda_1\|\boldsymbol{\theta}\|_1 + \tfrac{1}{2}\lambda_2\|\boldsymbol{\theta}\|_2^2 \tag{19}$$

The first step to finding the gradient is determining the differentiable space. Let the nonzero indices of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ be denoted $I(\boldsymbol{\lambda}) = \{i|\hat{\theta}_i(\boldsymbol{\lambda}) \neq 0 \text{ for } i = 1, ..., p\}$ and let $\boldsymbol{I}_{I(\boldsymbol{\lambda})}$ be a submatrix of the identity matrix with columns $I(\boldsymbol{\lambda})$. Since $|\cdot|$ is not differentiable at zero,

---
Algorithm 2 Gradient-based Joint Optimization

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

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

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

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

   Define the locally equivalent joint optimization problem

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

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

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

   with ${}_{\boldsymbol{U}^{(k)}}\nabla^2$ and ${}_{\boldsymbol{U}^{(k)}}\nabla$ are as defined in (14).

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

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

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

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

end for
---

the directional derivatives of $\|\boldsymbol{\theta}\|_1$ only exist along directions spanned by the columns of $\boldsymbol{I}_{I(\boldsymbol{\lambda})}$. Thus the differentiable space at $\boldsymbol{\lambda}$ is $S_{\boldsymbol{\lambda}} = span(\boldsymbol{I}_{I(\boldsymbol{\lambda})})$.

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

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \frac{1}{2}\|\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\|^2$$
$$\text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})}\boldsymbol{\beta}\|^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\lambda_2\|\boldsymbol{\beta}\|_2^2 \tag{20}$$

Since the problem is now smooth, we can apply the chain rule to get the gradient of the validation loss

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\lambda)}(\boldsymbol{X}_V)) = -\left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\frac{\partial}{\partial \boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right)^{\top} \left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\right) \tag{21}$$

and by (8),

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

2.4.2 Additive Models with Sparsity and Smoothness Penalties

Now consider the nonparametric regression problem given response y and covariates $\boldsymbol{x} \in \mathbb{R}^p$. We suppose y is the sum of p univariate functions:

$$y = \sum_{i=1}^{p} f_i(x_i) + \epsilon \tag{23}$$

Let $\boldsymbol{\theta}^{(i)} \equiv (f_i(x_{i1}), ..., f_i(x_{in}))$ be estimates of functions f_i at the observations. The model is fit using the least squares loss with sparsity and smoothness penalties. More specifically, for each estimate $\boldsymbol{\theta}^{(i)}$, we add a group lasso penalty $\|\cdot\|_2$ to encourage sparsity at the function level and a lasso penalty on the second-order discrete differences to encourage smooth function estimates. Hence there are a total of $2p$ non-smooth functions in the training criterion.

This regression problem usually employs two regularization parameters, one for the sum of the sparsity penalties and one for the sum of the smoothness penalties (Bühlmann & Van De Geer 2011). In the joint optimization problem below, we use a separate penalty parameter for each of the smoothness penalties, resulting in a total of $p + 1$ penalty parameters. Ideally the parameters could be tuned such that functions with nearly constant slope have large penalty parameters and "wiggly" functions have small penalty parameters.

Define matrices $\boldsymbol{I}_T$ and $\boldsymbol{I}_V$ such that $\boldsymbol{I}_T\boldsymbol{\theta}^{(i)}$ and $\boldsymbol{I}_V\boldsymbol{\theta}^{(i)}$ are estimates for f_i at the training and validation inputs, respectively. The joint optimization problem is

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}_+^{p+1}} \frac{1}{2|V|} \left\|\boldsymbol{y}_V - \boldsymbol{I}_V \sum_{i=1}^p \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda})\right\|_2^2$$

$$\text{s.t. } \hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left\|\boldsymbol{y}_T - \boldsymbol{I}_T \sum_{i=1}^p \boldsymbol{\theta}^{(i)}\right\|_2^2 + \lambda_0 \sum_{i=1}^p \|\boldsymbol{\theta}^{(i)}\|_2 \tag{24}$$

$$+\frac{1}{2}\sum_{i=1}^p \lambda_i \left\|\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}\boldsymbol{\theta}^{(i)}\right\|_1 + \frac{1}{2}\epsilon \sum_{i=1}^p \|\boldsymbol{\theta}^{(i)}\|_2^2$$

The differentiable space is straightforward to determine for this problem, though requires bulky notation. Define $I_i(\lambda)$ for $i = 1, ..., p$ to be the indices along which smoothness penalty is not differentiable

$$I_i(\lambda) = \left\{j| \left(\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}\hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda})\right)_j = 0 \text{ for } j = 1, ..., n\right\} \tag{25}$$

and

$$J(\lambda) = \left\{i|\hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) \neq \boldsymbol{0} \text{ for } i = 1, ..., p\right\} \tag{26}$$

The group lasso penalty $\|\cdot\|_2$ is not differentiable in any direction at $\boldsymbol{0}$ and is differentiable in all directions elsewhere. Then $S_{\boldsymbol{\lambda}} = span(\boldsymbol{U}^{(1)}) \oplus ... \oplus span(\boldsymbol{U}^{(p)})$ where $\boldsymbol{U}^{(i)} = \boldsymbol{0}$ if $\hat{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\lambda}) = \boldsymbol{0}$ and $\boldsymbol{U}^{(i)}$ is an orthonormal basis of $\mathcal{N}(\boldsymbol{I}_{I_i(\lambda)}\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)})$ otherwise.

Now we can define the locally equivalent joint optimization problem:

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}_+^{p+1}} \frac{1}{2|V|} \left\|\boldsymbol{y}_V - \boldsymbol{I}_V \sum_{i\in J(\boldsymbol{\lambda})} \boldsymbol{U}^{(i)}\hat{\boldsymbol{\beta}}^{(i)}(\boldsymbol{\lambda})\right\|_2^2$$

$$\text{s.t. } \hat{\boldsymbol{\beta}}^{(i)}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\left\|\boldsymbol{y}_T - \boldsymbol{I}_T \sum_{i\in J(\boldsymbol{\lambda})} \boldsymbol{U}^{(i)}\hat{\boldsymbol{\beta}}^{(i)}(\boldsymbol{\lambda})\right\|_2^2 + \lambda_0 \sum_{i\in J(\boldsymbol{\lambda})} \|\boldsymbol{U}^{(i)}\boldsymbol{\beta}^{(i)}\|_2 \tag{27}$$

$$+\frac{1}{2}\sum_{i\in J(\boldsymbol{\lambda})} \lambda_i \left\|\boldsymbol{D}_{\boldsymbol{x}_i}^{(2)}\boldsymbol{U}^{(i)}\boldsymbol{\beta}^{(i)}\right\|_1 + \frac{1}{2}\epsilon \sum_{i\in J(\boldsymbol{\lambda})} \|\boldsymbol{U}^{(i)}\boldsymbol{\beta}^{(i)}\|_2^2$$

From (8) and the chain rule, we get that the gradient of the validation loss is:

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) = -\frac{1}{|V|}\left(\boldsymbol{I}_V \sum_{i\in J(\boldsymbol{\lambda})} \boldsymbol{U}^{(i)}\frac{\partial}{\partial\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}^{(i)}(\boldsymbol{\lambda})\right)^\top \left(\boldsymbol{y}_V - \boldsymbol{I}_V \sum_{i\in J(\boldsymbol{\lambda})} \boldsymbol{U}^{(i)}\hat{\boldsymbol{\beta}}^{(i)}(\boldsymbol{\lambda})\right)$$

$$\tag{28}$$

where

$$\frac{\partial}{\partial\lambda_i}\begin{bmatrix} \hat{\boldsymbol{\beta}}^1(\boldsymbol{\lambda}) \\ ... \\ \hat{\boldsymbol{\beta}}^{|J(\boldsymbol{\lambda})|}(\boldsymbol{\lambda}) \end{bmatrix} = \boldsymbol{H}(\boldsymbol{\lambda})^{-1}\left[\begin{bmatrix} \frac{\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})}{\|\boldsymbol{U}^{(1)}\hat{\boldsymbol{\beta}}^{(1)}(\boldsymbol{\lambda})\|_2} \\ ... \\ \frac{\hat{\boldsymbol{\beta}}^{(M)}(\boldsymbol{\lambda})}{\|\boldsymbol{U}^{(p)}\hat{\boldsymbol{\beta}}^{(p)}(\boldsymbol{\lambda})\|_2} \end{bmatrix} \boldsymbol{C}(\boldsymbol{\beta}(\boldsymbol{\lambda}))\right] \tag{29}$$

The Hessian $\boldsymbol{H}(\boldsymbol{\lambda})$ and matrix $C(\boldsymbol{\beta}(\boldsymbol{\lambda}))$ are defined in the Appendix.

2.4.3 Un-pooled Sparse Group Lasso

The sparse group lasso is a linear regression problem that combines the $\|\cdot\|_2$ and $\|\cdot\|_1$ penalties (Simon et al. 2013). This method is well-suited for problems where features have a natural grouping, and only a few of the features from a few of the groups are thought to have an effect on response (e.g. genes in gene pathways). Here we consider a generalized version of sparse group lasso by considering individual penalty parameters for each group lasso penalty. This additional flexibility allows setting covariate and covariate group effects to zero by different thresholds. Hence "un-pooled" sparse group lasso may be better at modeling covariate groups with very different distributions.

The problem setup is as follows. Given M covariate groups, suppose $\boldsymbol{X}$ and $\boldsymbol{\theta}$ are partitioned into $\boldsymbol{X}^{(m)}$ and $\boldsymbol{\theta}^{(m)}$ for groups $m = 1, ..., M$. We are interested in finding the optimal regularization parameters $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, ..., \lambda_M)^\top$. The joint optimization problem is formulated as follows.

$$
\min_{\boldsymbol{\lambda} \in \mathbb{R}_+^{M+1}} \frac{1}{2|V|} \left\| \boldsymbol{y}_V - \boldsymbol{X}_V \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \right\|_2^2 \tag{30}
$$
$$
\text{s.t. } \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2|T|} \|\boldsymbol{y}_T - \boldsymbol{X}_T \boldsymbol{\theta}\|_2^2 + \lambda_0 \|\boldsymbol{\theta}\|_1 + \sum_{m=1}^{M} \lambda_m \|\boldsymbol{\theta}^{(m)}\|_2 + \frac{1}{2}\epsilon\|\boldsymbol{\theta}\|_2^2
$$

By the same logic as before, the differentiable space $S_{\boldsymbol{\lambda}}$ is $span(\boldsymbol{I}_{I(\boldsymbol{\lambda})})$ where $I(\boldsymbol{\lambda}) = \{i | \hat{\theta}_i(\boldsymbol{\lambda}) \neq 0$ for $i = 1, ..., p\}$ are the nonzero indices of $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$. Therefore the locally equivalent joint optimization problem is

$$
\min_{\boldsymbol{\lambda} \in \mathbb{R}_+^{M+1}} \frac{1}{2|V|} \left\| \boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right\|_2^2 \tag{31}
$$
$$
\text{s.t. } \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2|T|} \left\| \boldsymbol{y}_T - \boldsymbol{X}_{T,I(\boldsymbol{\lambda})} \boldsymbol{\beta} \right\|_2^2 + \lambda_0 \|\boldsymbol{\beta}\|_1 + \sum_{m=1}^{M} \lambda_m \|\boldsymbol{\beta}^{(m)}\|_2 + \frac{1}{2}\epsilon\|\boldsymbol{\beta}\|_2^2
$$

where we use the same notational shorthand $\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}$ and $\boldsymbol{X}_{V,I(\boldsymbol{\lambda})}$ from Section 2.4.1.

From (8) and the chain rule, we get that the gradient of the validation loss is

$$
\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{y}_V, f_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}(\boldsymbol{X}_V)) = -\frac{1}{n} \left(\boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right)^\top \left(\boldsymbol{y}_V - \boldsymbol{X}_{V,I(\boldsymbol{\lambda})} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \right) \tag{32}
$$

where

$$
\frac{\partial}{\partial \boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\boldsymbol{H}(\boldsymbol{\lambda})^{-1} \left[\boldsymbol{C}(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \quad sgn(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})) \right] \tag{33}
$$

The Hessian $\boldsymbol{H}(\boldsymbol{\lambda})$ and the matrix $\boldsymbol{C}(\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}))$ are given in the Appendix.

2.4.4 Low-Rank Matrix Completion

In this example, we move away from the simple regression framework and consider matrix-valued data with partially observed entries. The goal in matrix-completion problems is to reconstruct the rest of the matrix. There are many applications where such problems come up, including collaborative filtering SIGKDD & Netflix (2007) and robust principle components analysis Candès et al. (2011). These problems often try to find low-rank matrix representations by minimizing the nuclear norm $\| \cdot \|_*$ (Fazel 2002, Srebro 2004). The nuclear norm is non-smooth since it is the ℓ_1 norm of the singular values.

In this example, we have a matrix-completion problem with additional information in the form of feature vectors for the rows and columns. We suppose that the matrix is the sum of row and column effects as well as an interaction matrix. Suppose the row and column effects are linear functions of the row and column feature vectors, respectively. (Fithian & Mazumder 2013) solves this problem with three penalty parameters – one for the nuclear norm on the interaction matrix and one for each of the penalty functions on the row and column coefficients. Here we have row and column feature groups, and we would like to recover a sparse set of groups using group lasso penalties.

The exact problem setup is as follows. Suppose we are performing matrix-completion for the matrix $M \in \mathbb{R}^{N \times N}$. Let $X \in \mathbb{R}^{N \times p}$ be the feature vectors for the rows and $Z \in \mathbb{R}^{N \times p}$ be the feature vectors for columns. Suppose the matrix is the sum of linear functions of the row and column features, interaction matrix Γ, and noise ϵ:

$$M_{ij} = x_i \alpha + z_j \beta + \Gamma_{ij} + \epsilon_{ij} \tag{34}$$

Let the observed matrix positions in training and validation sets be denoted T and V. $\| \cdot \|_T^2$ denotes the Frobenius norm over the entries in T and similarly for $\| \cdot \|_V^2$. In addition, suppose the row and column features are composed of G groups. We partition the row and column coefficient vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ into $\boldsymbol{\alpha}^{(g)}$ and $\boldsymbol{\beta}^{(g)}$ for groups $g = 1, ..., G$.

We are interested in finding the optimal regularization parameters $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, ..., \lambda_{2G})^\top$.

The joint optimization problem is formulated as follows.

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}_+^{2G+1}} \frac{1}{2|V|}\left\|\boldsymbol{M}-\boldsymbol{X}\hat{\boldsymbol{\alpha}}(\boldsymbol{\lambda})\mathbf{1}^\top-(\boldsymbol{Z}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\mathbf{1}^\top)^\top-\hat{\boldsymbol{\Gamma}}(\boldsymbol{\lambda})\right\|_V^2$$

$$\text{s.t. } \hat{\boldsymbol{\alpha}}(\boldsymbol{\lambda}),\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}),\hat{\boldsymbol{\Gamma}}(\boldsymbol{\lambda})=\arg\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\Gamma}}\frac{1}{2|T|}\left\|\boldsymbol{M}-\boldsymbol{X}\hat{\boldsymbol{\alpha}}(\boldsymbol{\lambda})\mathbf{1}^\top-(\boldsymbol{Z}\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})\mathbf{1}^\top)^\top-\boldsymbol{\Gamma}\right\|_T^2 \quad (35)$$

$$+\lambda_0\|\boldsymbol{\Gamma}\|_*+\sum_{g=1}^G\lambda_g\|\boldsymbol{\alpha}^{(g)}\|_2+\sum_{g=1}^G\lambda_{g+G}\|\boldsymbol{\beta}^{(g)}\|_2+\tfrac{1}{2}\epsilon\left(\|\boldsymbol{\alpha}\|_2^2+\|\boldsymbol{\beta}\|_2^2+\|\boldsymbol{\Gamma}\|_F^2\right)$$

The differentiable space for $\boldsymbol{\alpha}$ is $span(\boldsymbol{I}_{I_r(\boldsymbol{\lambda})})$ where

$$\boldsymbol{I}_r(\boldsymbol{\lambda})=span(\{i|\text{index } i \text{ in } \boldsymbol{\alpha}^{(g)} \text{ where } \hat{\boldsymbol{\alpha}}^{(g)}(\boldsymbol{\lambda})\neq\boldsymbol{0} \text{ for } g=1,...,G\})$$

The differentiable space for $\boldsymbol{\beta}$, denoted $span(\boldsymbol{I}_{I_c(\boldsymbol{\lambda})})$, is defined similarly. The differentiable space for Γ is the set of all matrices that have the same rank. This follows from the fact that the nuclear norm is a lasso penalty over the singular values. For more details, refer to the appendix. Putting this together, we have

$$S_{\boldsymbol{\lambda}}=\{\Gamma:rank(\Gamma)=rank(\hat{\boldsymbol{\Gamma}}(\boldsymbol{\lambda}))\}\oplus span(\boldsymbol{I}_{I_r(\boldsymbol{\lambda})})\oplus span(\boldsymbol{I}_{I_c(\boldsymbol{\lambda})}) \quad (36)$$

The locally equivalent joint optimization problem is therefore

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}_+^{2G+1}} \frac{1}{2|V|}\left\|\boldsymbol{M}-\boldsymbol{X}_{I_r(\boldsymbol{\lambda})}\hat{\boldsymbol{\eta}}(\boldsymbol{\lambda})\mathbf{1}^\top-(\boldsymbol{Z}_{I_c(\boldsymbol{\lambda})}\hat{\boldsymbol{\gamma}}(\boldsymbol{\lambda})\mathbf{1}^\top)^\top-\hat{\boldsymbol{\Xi}}(\boldsymbol{\lambda})\right\|_V^2$$

$$\text{s.t. } \hat{\boldsymbol{\eta}}(\boldsymbol{\lambda}),\hat{\boldsymbol{\gamma}}(\boldsymbol{\lambda}),\hat{\boldsymbol{\Xi}}(\boldsymbol{\lambda})=\arg\min_{\boldsymbol{\eta},\boldsymbol{\gamma},\boldsymbol{\Xi}}\frac{1}{2|T|}\left\|\boldsymbol{M}-\boldsymbol{X}_{I_r(\boldsymbol{\lambda})}\hat{\boldsymbol{\eta}}(\boldsymbol{\lambda})\mathbf{1}^\top-(\boldsymbol{Z}_{I_c(\boldsymbol{\lambda})}\hat{\boldsymbol{\gamma}}(\boldsymbol{\lambda})\mathbf{1}^\top)^\top-\boldsymbol{\Xi}\right\|_T^2$$

$$+\lambda_0\|\boldsymbol{\Xi}\|_*+\sum_{g=1}^G\lambda_g\|\boldsymbol{\eta}^{(g)}\|_2+\sum_{g=1}^G\lambda_{G+g}\|\boldsymbol{\gamma}^{(g)}\|_2+\tfrac{1}{2}\epsilon\left(\|\boldsymbol{\eta}\|_2^2+\|\boldsymbol{\gamma}\|_2^2+\|\boldsymbol{\Xi}\|_F^2\right)$$

$$\text{and } rank(\boldsymbol{\Xi})=rank(\hat{\boldsymbol{\Gamma}}(\boldsymbol{\lambda}))$$

$$(37)$$

where we use the same notational shorthand $\boldsymbol{X}_{T,I(\boldsymbol{\lambda})}$ and $\boldsymbol{X}_{V,I(\boldsymbol{\lambda})}$ from Section 2.4.1.

To calculate the gradient of the validation loss, we need to slightly modify the steps outlined in Algorithm 2. The details are given in the Appendix.

2.5 Gradient Descent Details

Here we discuss choice of step size and our convergence criterion.

There are many possible choices for our step size sequence $\{t^{(k)}\}$. Popular choices for convex problems are discussed in Boyd & Vandenberghe (2004). We chose a backtracking line search as discussed in Chapter 9. In our examples initial step size was 1, and we backtrack with parameters $\alpha=0.001$ and $\beta=0.1$. Details of backtracking line search are given in the

15

Appendix. During gradient descent, it is possible that the step size will result in a negative regularization parameter; we reject any step that would set a regularization parameter to below a minimum threshold of 1e-6.

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

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

for some prespecified tolerance δ. For the results in this manuscript we use $\delta = 0.0005$.

3   Simulation Studies

We now compare our gradient descent algorithm to gradient-free methods through simulation studies. Each simulation corresponds to a joint optimization problem given in Section 2.4. We tune the regularization parameters over a training/validation split using gradient descent, Nelder-Mead, and the Bayesian optimization solver from Snoek et al. (2012) called Spearmint. For baseline comparison, we also solve a two-parameter version of the joint optimization problem using gradient descent, Nelder-Mead, Spearmint, and grid search. If the simulation involved tuning more than forty penalty parameters, we only used gradient descent to tune the parameters. Gradient-free methods are generally not recommended for tuning more than thirty parameters; Spearmint's implementation is limited to no more than forty hyperparameters and Nelder-Mead performed poorly.

We compare the efficiency of the methods by the number of times the methods solved the inner training criterion (labeled "# Solves" in the tables below). We set this number to 100 for all three gradient-free methods. This number is variable for gradient descent since it depends on how fast the algorithm converges.

There are two computational concerns when tuning regularization parameters by gradient descent. First, the gradient calculation can be slow if the matrix in (9) is large. Bengio (2000) and Foo et al. (2008) suggest backpropagating through a Cholesky decomposition or using conjugate gradients to speed this up. This is actually not an issue for problems with non-smooth training criteria since the solutions tend to be sparse, which result in small linear systems. The second concern is that the inner optimization problem must be solved to a high

Table 1: Comparison of solvers for Elastic Net joint optimization problem. Standard errors are given in parentheses.

	Validation Error	Test Error	# Solves
Gradient Descent	4.92 (0.41)	5.44 (0.20)	32.40
Nelder-Mead	4.87 (0.40)	5.50 (0.19)	100
Spearmint	5.15 (0.41)	5.76 (0.22)	100
Grid Search	5.15 (0.44)	5.47 (0.19)	100

accuracy in order to calculate the gradient. (Recall that the gradient is derived via implicit differentiation.) To address this, we allow more iterations for solving the inner optimization problem. For a faster implementation of gradient descent, one can use a more specialized solver.

In each section below, we provide the simulation settings followed by a discussion of the results. Each simulation was run thirty times.

3.1 Elastic Net

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

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \sigma\boldsymbol{\epsilon} \text{ where } \boldsymbol{\beta} = (\underbrace{1, ..., 1}_{\text{size } 15}, \underbrace{0, ..., 0}_{\text{size } 235}) \tag{38}$$

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

Grid search was performed over a 10×10 log-spaced grid from $1e$-5 to 100. Nelder-mead and gradient descent were initialized at (0.01, 0.01) and (10, 10). Nelder-mead was allowed fifty iterations starting from each initialization point.

As shown in Table 1, all the methods achieve similar validation errors. For this simple problem, the benefit for using gradient descent is not significant.

3.2 Additive model with Smoothness and Sparsity Penalty

Each dataset consists of 100 training, 50 validation, and 50 test observations with $p = 23$ covariates. The covariates $\boldsymbol{x_i} \in \mathbb{R}^n$ for $i = 1, ...p$ are equally spaced from -5 to 5 with a random displacement $\delta \sim U(0, \frac{1}{300})$ at the start and then shuffled randomly. The true model is the sum of three nonzero functions and 20 zero functions. Response y was generated as follows

$$
\begin{aligned}
\boldsymbol{y} &= \sum_{i=1}^{p} \boldsymbol{f}_i(\boldsymbol{x}_i) + \sigma \boldsymbol{\epsilon} \\
f_1(x) &= 9 \sin(3x) \\
f_2(x) &= x \\
f_3(x) &= 6 \cos(1.25x) + 6 \sin(0.5x + 0.5) \\
f_i(x) &= 0 \text{ for } i = 4, ..., 23
\end{aligned}
$$

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

Nelder-Mead and gradient descent were both initialized at $(10, 1, ..., 1)$ and $(0.1, 0.01, ..., 0.01)$. Nelder-mead was allowed fifty iterations starting from each initialization point.

For baseline comparison, we also solved a two-parameter version of this joint optimization problem by pooling $\{\lambda_i\}_{i=1:p}$ into a single λ_1. Grid search was then performed over a 10×10 log-spaced grid from $1e$-4 to 100.

As shown in Table 2, gradient descent returned models with the lowest test error on average. We can see that the joint optimization problem with 24 penalty parameters is sensitive to the penalty parameters chosen; Spearmint returned a result that is not only worse than gradient descent but even worse than grid search over the pooled two-parameter joint optimization problem.

Table 3 provides the average penalty parameter values for $\lambda_0, ..., \lambda_4$. Gradient descent was indeed able to determine the smoothness of the functions. It tended to choose λ_1 to have the smallest value and λ_2 to have the largest value, which corresponds to the fact that the slope for f_1 varies the most and the slope for f_2 is constant. In contrast, Nelder-Mead wasn't able to determine the difference in smoothness between the functions and kept all penalty parameters the same. Spearmint chose a very different set of parameters that don't seem to be appropriate for the problem.

Table 2: A comparison of additive models with separate smoothness penalty parameters tuned by gradient descent, Nelder-Mead, and Spearmint vs. additive models with two penalty parameters tuned by grid search. Standard errors are given in parentheses.

	Num λ	Validation Error	Test Error	# Solves
Gradient Descent	24	23.87 (0.97)	26.10 (0.86)	13.07
Nelder-Mead	24	27.94 (0.90)	28.73 (0.90)	100
Spearmint	24	27.91 (1.80)	34.02 (1.59)	100
Grid Search	2	28.71 (0.97)	29.42 (0.96)	100

Table 3: Average λ_i values for the fitted models

	λ_0	λ_1	λ_2	λ_3	λ_4
Gradient Descent	9.95	0.40	1.01	0.86	1.04
Nelder-Mead	9.76	0.90	1.00	1.00	1.01
Spearmint	0.026	0.003	0.02	0.006	0.018
Grid Search	4.04	0.38	–	–	–

Figure 1 provides example model fits given by gradient descent. The smoothness of the function estimates reflect the magnitude of the regularization parameters given in Table 2. In addition, gradient descent was able to learn that f_4 was the zero function.

3.3 Un-pooled Sparse group lasso

We ran three experiments with different numbers of covariate groups M and total covariates p, as given in Table 4. For each experiment, the dataset consisted of n training, $n/3$ validation, and 200 test observations. The predictors $\boldsymbol{X}$ were generated from a standard normal distribution. The response $\boldsymbol{y}$ was generated by

$$\boldsymbol{y} = \sum_{j=1}^{3} \boldsymbol{X}^{(j)} \boldsymbol{\beta}^{(j)} + \sigma \boldsymbol{\epsilon} \text{ where } \boldsymbol{\beta}^{(j)} = (1, 2, 3, 4, 5, 0, ..., 0) \tag{39}$$

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

We compare gradient descent against all three gradient-free methods in the first experiment with 31 regularization parameters and only compare against grid search for the latter

Figure 1: Example model fits given by gradient descent for f_1, f_2, f_3, and f_4. The dashed lines are the true functions and the solid lines are the estimated functions. (Top left: f_1, top right: f_2, bottom left: f_3, bottom right: f_4)
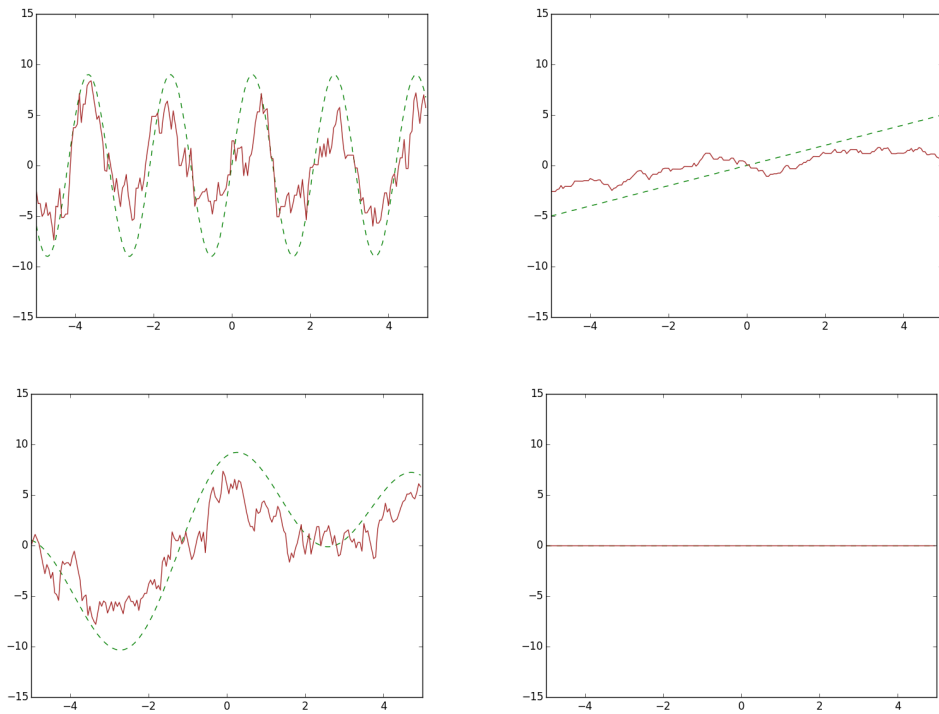
experiments with 61 and 101 regularization parameters. For all experiments, grid search solved a two-parameter version where $\{\lambda_i\}_{i=1:M}$ are pooled into a single λ_1.

For all three experiments, grid search was performed over a 10×10 grid from $1e\text{-}3$ to 10. Gradient descent was initialized at $0.1 \times \mathbf{1}$ and $\mathbf{1}$. Nelder-Mead was initialized at the same points for the first experiment.

Model performance was assessed by the test error and β error (defined as $\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2$). As shown in Table 4, the model tuned by gradient descent produced the lowest test error and β error in all three experiments. Nelder-Mead struggled to minimize the validation error and its test error was similar to that from grid search for the two penalty-parameter problem. Interestingly, Spearmint found models with small validation error but had the highest test error.

The validation errors of the un-pooled sparse group lasso models fit by gradient descent were much smaller than the test errors. This behavior can be attributed to the larger model space when we increase the number of penalty parameters. If we think of this within the bias-variance framework, the un-pooled sparse group lasso has smaller bias but larger variance. In these experiments, we find that the improvement in bias outweighs the additional variance; that is, the drop in validation error is larger than the gap between the validation and test error. Comparing the results from gradient descent to Nelder-mead and Spearmint, we see that it is crucial to apply methods that can effectively minimize the validation error. Otherwise the performance of the fitted model can be worse than a two-penalty-parameter version of the problem.

3.4   Low-Rank Matrix Completion

For this experiment, we considered a 60×60 matrix. The training set contained roughly two observed entries from each row and column and the validation set contained one per row and column. The row and column feature vectors contained 12 covariate groups each, where each group contained three covariates.

The true coefficients are $\boldsymbol{\alpha}^{(g)} = g\mathbf{1}$ for $g = 1, ..., 4$ and $\boldsymbol{\beta}^{(g)} = g\mathbf{1}$ for $g = 1, 2$. The rest of the coefficients were zero. The interaction matrix was rank one and set to be $\boldsymbol{u}\boldsymbol{v}^\top$ where $\boldsymbol{u}$ and $\boldsymbol{v}$ were sampled from a standard normal distribution. The predictors $\boldsymbol{X}$ and $\boldsymbol{Z}$ were

Table 4: Un-pooled sparse group lasso and sparse group lasso tuned by gradient descent and grid search, respectively. Standard errors are given in parentheses. We abbreviated the methods as follows: Gradient Descent = GD, Nelder-Mead = NM, Spearmint = SP, Grid Search = GS

n=90, p=600, M=30					
	# λ	β Error	Validation Err	Test Err	# Solves
GD	31	5.93 (0.23)	18.88 (0.78)	38.92 (1.47)	40.43 (0.64)
NM	31	7.41 (0.17)	47.78 (2.25)	49.45 (1.43)	100
SP	31	7.85 (0.15)	25.91 (1.44)	53.18 (1.54)	100
GS	2	7.43 (0.19)	47.23 (2.26)	50.01 (1.40)	100
n=90, p=900, M=60					
	# λ	β Error	Validation Error	Test Error	# Solves
GD	61	6.43 (0.23)	18.78 (1.25)	41.88 (1.51)	38.13 (1.48)
GS	2	7.67 (0.20)	45.70 (2.27)	51.34 (1.86)	100
n=90, p=1200, M=100					
	# λ	β Error	Validation Error	Test Error	# Solves
GD	101	6.93 (0.26)	17.91 (1.44)	47.47 (2.00)	37.83 (1.33)
GS	2	8.28 (0.19)	50.00 (2.16)	57.14 (2.18)	100

Table 5: Matrix Completion. Standard errors are given in parentheses. We abbreviated the methods as follows: Gradient Descent = GD, Nelder-Mead = NM, Spearmint = SP, Grid Search = GS

	# λ	Validation Err	Test Err	Num Solves
GD	25	?		
NM	25	?	100	
SP	25	?	100	
GS	2	?	100	

sampled from a standard normal distribution and scaled so the l_2 norm of $\boldsymbol{X\alpha}\mathbf{1}^\top + \boldsymbol{Z\beta}\mathbf{1}^\top$ was the same as $\boldsymbol{\Gamma}$. The noise $\boldsymbol{\epsilon}$ was generated from a standard normal distribution and scaled such that the signal to noise ratio was 2.

Grid search tuned the two-parameter version of (35) where $\{\lambda_i\}_{i=1:2G}$ are pooled into a single λ_1. Grid search was performed over a 10×10 grid from $1e\text{-}3.5$ to -1. Gradient descent was initialized at $0.005 \times \mathbf{1}$ and $0.003 \times \mathbf{1}$. Nelder-Mead was initialized at the same points.

As seen in Table 5, gradient descent had the lowest average validation and test error.

4 Application to Biological Data

Finally, we applied our algorithm in a real data example. More specifically, we considered the problem of finding predictive genes from gene pathways for Crohn's Disease and Ulcerative Colitis. Simon et al. (2013) addressed this problem using the sparse group lasso; we now compare this against applying the un-pooled sparse group lasso, where the regularization parameters were tuned using gradient descent. Since this is a classification task, the joint optimization problem is the same as (30) but with the logistic loss:

$$L\left(\boldsymbol{y}, f_{\boldsymbol{\beta}(\boldsymbol{\lambda})}(\boldsymbol{X})\right) = \sum_{i=1}^{n} y_i \log\left(\frac{1}{1 + \exp(-\boldsymbol{x}_i^\top \boldsymbol{\beta})}\right) + (1 - y_i) \log\left(1 - \frac{1}{1 + \exp(-\boldsymbol{x}_i^\top \boldsymbol{\beta})}\right) \quad (40)$$

Our dataset is from a colitis study of 127 total patients, 85 with colitis (59 crohn's patients + 26 ulcerative colitis patients) and 42 healthy controls (Burczynski et al. 2006). Expression data was measured for 22,283 genes on affymetrix U133A microarrays. We grouped the genes

Table 6: Predictive genes and genesets of Ulcerative Colitis found by un-pooled sparse group lasso vs. sparse group lasso. Standard errors given in parenthesis. FP = False positive, FN = False negative

	# λ	% FP	% FN	% Correct	# Genesets	# Genes
Gradient Descent	327	3.72 (1.12)	19.90 (2.30)	89.48 (.93)	12.10 (0.98)	55.60 (6.92)
Grid Search	2	3.02 (0.57)	25.34 (2.94)	86.88 (1.65)	30.80 (5.18)	215.40 (20.69)

according to the 326 C1 positional gene sets from MSigDb v5.0 (Subramanian et al. 2005) and discarded the 2358 genes not found in the gene set.

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

Table 6 presents the average results from repeating this process ten times. Un-pooled sparse group lasso achieved a slightly higher classification rate than sparse group lasso. Interestingly, un-pooled sparse group lasso found solutions that were significantly more sparse than sparse group lasso; on average, un-pooled sparse group lasso identified 9 genesets whereas sparse group lasso identified 38. These results suggest that un-pooling the penalty parameters in sparse group lasso could potentially improve interpretability.

5   Discussion

In this paper we showed how to calculate the exact gradient for joint optimization problems with non-smooth penalty functions that are smooth almost everywhere. In addition we provide an algorithm for tuning the regularization parameters using a variation of gradient descent.

The simulation studies show that for certain problems, separating the penalty parameters can improve model performance. However, it is crucial that the regularization parameters be tuned appropriately. For the same joint optimization problem with many penalty parameters, gradient descent is able to return good model fits whereas gradient-free methods like Nelder-

Mead and Bayesian optimization tend to fail. In fact, when the optimization method is unable to tune the regularization parameters appropriately, we find that a simple grid search over the pooled two-parameter joint optimization problem can result in better models.

Also, gradient descent tends to require a fewer number of times to solve the inner optimization problem. This makes the method quite appealing for problems that require large solve times.

Through our simulation studies, we did find that this gradient-based approach depends on solving the inner optimization problem to a higher level of accuracy than needed for gradient-free approaches. More work could be done to investigate the degree of accuracy required for gradient descent to still be effective.

Finally, an open theoretical question is how much the model complexity increases when too many penalty parameters are introduced. Similar to how model parameters can overfit to their training data, it is possible for the penalty parameters to overfit to the training and validation data.

References

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

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

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

Bühlmann, P. & Van De Geer, S. (2011), *Statistics for high-dimensional data: methods, theory and applications*, Springer Science & Business Media.

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

Candès, E. J., Li, X., Ma, Y. & Wright, J. (2011), 'Robust principal component analysis?', *Journal of the ACM (JACM)* **58**(3), 11.

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

Fazel, M. (2002), Matrix rank minimization with applications, PhD thesis, PhD thesis, Stanford University.

Fithian, W. & Mazumder, R. (2013), 'Scalable convex methods for flexible low-rank matrix modeling', *arXiv preprint arXiv:1308.4211* .

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

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

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

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

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

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

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

Neal, R. M. (1996), 'Bayesian learning for neural networks'.

Nelder, J. A. & Mead, R. (1965), 'A simplex method for function minimization', *The computer journal* **7**(4), 308–313.

SIGKDD, A. & Netflix (2007), 'Soft modelling by latent variables: the nonlinear iterative partial least squares (nipals) approach', *Proceedings of KDD Cup and Workshop* .

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

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

Srebro, N. (2004), Learning with matrix factorizations, PhD thesis, Citeseer.

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

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

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

van de Geer, S. & Muro, A. (2014), 'The additive model with different smoothness for the components', *arXiv preprint arXiv:1405.6584* .

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

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