# Optimizing Approximate Leave-one-out Cross-validation to Tune Hyperparameters

**Ryan Burn** ryan.burn@gmail.com
*Department of Mathematics*
*University of Washington*
*Seattle, WA 98195-4350, USA*

**Editor:**

## Abstract

For a large class of regularized models, leave-one-out cross-validation can be efficiently estimated with an approximate leave-one-out formula (ALO). We consider the problem of adjusting hyperparameters so as to optimize ALO. We derive efficient formulas to compute the gradient and hessian of ALO and show how to apply a second-order optimizer to find hyperparameters. We demonstrate the usefulness of the proposed approach by finding hyperparameters for regularized logistic regression and ridge regression on various real-world data sets.

**Keywords:** hyperparameter optimization, regularization, logistic regression, ridge regression, trust-region methods

## 1. Introduction

Let $\mathcal{D} = \{(y_1, \boldsymbol{x}_1), \cdots, (y_n, \boldsymbol{x}_n)\}$ denote a data set, where $\boldsymbol{x}_i \in \mathbb{R}^p$ are features and $y_i \in \mathbb{R}$ are responses. In many applications, we model the observations as independent and identically distributed draws from an unknown joint distribution of the form $q(y_i, \boldsymbol{x}_i) = q_1\left(y_i \mid \boldsymbol{x}_i^\top \boldsymbol{\beta}_*\right) q_2(\boldsymbol{\beta}_*)$, and we estimate $\boldsymbol{\beta}_*$ using the optimization problem:

$$\hat{\boldsymbol{\beta}} \triangleq \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \ell\left(y_i \mid \boldsymbol{x}_i^\top \boldsymbol{\beta}\right) + R_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) \right\}, \tag{1}$$

where $\ell$ is a loss function, $R_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) = \sum_{j=1}^{p} r_{j,\boldsymbol{\lambda}}(\hat{\beta}_j)$ is a regularizer, and $\boldsymbol{\lambda}$ represents hyperparameters. $R_{\boldsymbol{\lambda}}$ controls the complexity of the model by penalizing larger values of $\boldsymbol{\beta}$ and can prevent overfitting. We aim to select $\boldsymbol{\lambda}$ so as to minimize the out-of-sample prediction error ($\mathrm{Err}_{\mathrm{out}}$):

$$\mathrm{Err}_{\mathrm{out}} \triangleq \mathbb{E}\left[\ell\left(y_o \mid \boldsymbol{x}_o^\top \hat{\boldsymbol{\beta}}\right) \mid \mathcal{D}\right],$$

where $(y_o, \boldsymbol{x}_o)$ is an unseen sample from the distribution $q(y, \boldsymbol{x})$. Because $q$ is unknown, we estimate $\mathrm{Err}_{\mathrm{out}}$ with a function $f(\boldsymbol{\lambda} \mid \mathcal{D})$ and apply a hyperparameter optimization algorithm to find

$$\hat{\boldsymbol{\lambda}} \triangleq \underset{\boldsymbol{\lambda}}{\operatorname{argmin}}\, f(\boldsymbol{\lambda} \mid \mathcal{D}).$$

Empirical evidence has shown leave-one-out cross-validation (LO) to be an accurate method for estimating $\text{Err}_{\text{out}}$ (Rad et al., 2020). In general, LO can be expensive to compute, requiring a model to be fit $n$ times; however, under certain conditions, it can be efficiently estimated using a closed-form approximate leave-one-out formula (ALO), and the approximation has been shown to be accurate in high-dimensional settings (Rad and Maleki, 2020). In addition, for the special case of ridge regression, ALO is exact and equivalent to Allen's PRESS (Allen, 1974).

In this paper, we derive efficient formulas to compute the gradient and hessian of ALO, given a smooth loss function and a smooth regularizer, and we show how to apply a trust-region algorithm for optimization to find hyperparameters for several different classes of regularized models.

### 1.1 Relevant Work

Grid search is a commonly used approach for hyperparameter optimization. Although it can work well for models with only a single hyperparameter, it requires a search space to be specified and quickly becomes inefficient when multiple hyperparameters are used (Bergstra and Bengio, 2012). Other scalable gradient-based approaches have focused on using holdout sets to approximate $\text{Err}_{\text{out}}$ (Do et al., 2008; Bengio, 2000). Using ALO, we expect our objective function to be a more accurate proxy for $\text{Err}_{\text{out}}$. We further improve on previous approaches by computing the hessian of our objective function and applying a trust-region algorithm for optimization, allowing us to make global convergence guarantees.

### 1.2 Notation

We denote vectors by lowercase bold letters and matrices by uppercase bold letters. We use the notation $\text{vec}\left[\{a_i\}_i\right]$ to denote the column vector $(a_1, a_2, \ldots)^\top$. For a function $f\colon \mathbb{R} \to \mathbb{R}$, we denote its 1st, 2nd, 3rd, and 4th derivatives by $\dot{f}$, $\ddot{f}$, $\dddot{f}$, and $\ddddot{f}$, respectively. Finally, given a function $f\colon \mathbb{R}^p \to \mathbb{R}$, we denote the gradient by $\nabla f$ and the hessian by $\nabla^2 f$.

## 2. Preliminary: Trust-region Methods

Let $f\colon \mathbb{R}^p \to \mathbb{R}$ denote a twice-differentiable objective function. Trust-region methods are iterative, second-order optimization algorithms that produces a sequence $\{\boldsymbol{x}_k\}$, where the $k^{\text{th}}$ iteration is generated by updating the previous iteration with a solution to the subproblem (Sorensen, 1982):

$$\begin{aligned}
\boldsymbol{x}_k &= \boldsymbol{x}_{k-1} + \hat{\boldsymbol{s}}_k \quad \text{and} \\
\hat{\boldsymbol{s}}_k &= \underset{\boldsymbol{s}}{\text{argmin}} \left\{ f\left(\boldsymbol{x}_{k-1}\right) + \nabla f\left(\boldsymbol{x}_{k-1}\right)^\top \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^\top \nabla^2 f\left(\boldsymbol{x}_{k-1}\right) \boldsymbol{s} \right\} \\
&\qquad \text{s.t.} \quad \|\boldsymbol{s}\| \leq \delta_k.
\end{aligned}$$

The subproblem minimizes the second-order approximation of $f$ at $\boldsymbol{x}_{k-1}$ within the neighborhood $\|\boldsymbol{s}\| \leq \delta_k$, called the trust region. Using the trust region, we can restrict the second-order approximation to areas where it models $f$ well. Efficient algorithms exist to solve the subproblem regardless of whether $\nabla^2 f(\boldsymbol{x}_{k-1})$ is positive-definite, making trust-region methods well-suited for non-convex optimization problems (Moré and Sorensen, 1983). With

proper rules for updating $\delta_k$ and standard assumptions, such as Lipschitz continuity of $\nabla f$, trust-region methods are globally convergent. Moreover, if $\nabla^2 f$ is Lipschitz continuous for all $\boldsymbol{x}$ sufficiently close to a nondegenerate second-order stationary point $\boldsymbol{x}_*$, where $\nabla^2 f(\boldsymbol{x}_*)$ is positive-definite, then trust-region methods have quadratic local convergence (Nocedal and Wright, 1999).

## 3. Preliminary: ALO

Put $\ell_i(u) \triangleq \ell(y_i \mid u)$ and $u_i = \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}$. The LO estimate for $\mathrm{Err}_{\mathrm{out}}$ is defined as

$$\mathrm{LO}_{\boldsymbol{\lambda}} \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell_i \left( \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}_{/i} \right),$$

where

$$\hat{\boldsymbol{\beta}}_{/i} \triangleq \operatorname*{argmin}_{\boldsymbol{\beta}} \left\{ \sum_{j \neq i} \ell_j \left( \boldsymbol{x}_j^\top \boldsymbol{\beta} \right) + R_{\boldsymbol{\lambda}} \left( \boldsymbol{\beta} \right) \right\}. \tag{2}$$

ALO works by finding $\hat{\boldsymbol{\beta}}$ and then using Newton's method to approximate $\hat{\boldsymbol{\beta}}_{/i}$ from the gradient and hessian of Equation 2 at $\hat{\boldsymbol{\beta}}$. Let $\boldsymbol{g}_{/i}$ and $\boldsymbol{H}_{/i}$ denote the gradient and hessian, respectively, of Equation 2 at $\hat{\boldsymbol{\beta}}$. The ALO approximation to $\hat{\boldsymbol{\beta}}_{/i}$ is $\tilde{\boldsymbol{\beta}}_{/i} \triangleq \hat{\boldsymbol{\beta}} - \boldsymbol{H}_{/i}^{-1} \boldsymbol{g}_{/i}$, and

$$\mathrm{ALO}_{\boldsymbol{\lambda}} \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell_i \left( u_i - \boldsymbol{x}_i^\top \boldsymbol{H}_{/i}^{-1} \boldsymbol{g}_{/i} \right).$$

Computed naively, this formula would require solving a linear system of order $p$ $n$ times, but we can achieve a more efficient form by applying the matrix inversion lemma. Let $\boldsymbol{X}$ represent the feature matrix whose i$^{\text{th}}$ row is $\boldsymbol{x}_i$; let $\boldsymbol{H}$ denote the hessian of Equation 1 at $\hat{\boldsymbol{\beta}}$. Define $\boldsymbol{W} \triangleq \nabla^2 R_{\boldsymbol{\lambda}}(\hat{\boldsymbol{\beta}})$. Then,

$$\boldsymbol{H} = \boldsymbol{X}^\top \boldsymbol{A} \boldsymbol{X} + \boldsymbol{W},$$

where $\boldsymbol{A}$ is a diagonal matrix with $\boldsymbol{A}_{ii} = \ddot{\ell}_i \left( u_i \right)$, and

$$\mathrm{ALO}_{\boldsymbol{\lambda}} = \frac{1}{n} \sum_{i=1}^{n} \ell_i \left( u_i + \frac{\dot{\ell}_i \left( u_i \right) h_i}{1 - \ddot{\ell}_i \left( u_i \right) h_i} \right),$$

where $h_i \triangleq \boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \boldsymbol{x}_i$. See Rad and Maleki (2020) for a derivation.

## 4. Optimizing ALO

Put

$$\tilde{u}_{/i} \triangleq u_i + \frac{\dot{\ell}_i \left( u_i \right) h_i}{1 - \ddot{\ell}_i \left( u_i \right) h_i} \quad \text{and} \quad f \left( \boldsymbol{\lambda} \right) \triangleq \sum_{i=1}^{n} \ell_i(\tilde{u}_{/i}).$$

Minimizing $f$ is equivalent to minimizing $\mathrm{ALO}_{\boldsymbol{\lambda}}$. We present formulas for computing $\nabla f$ and $\nabla^2 f$. See Appendix A and Appendix B for derivations. In general, $\nabla^2 f$ will not be positive-definite; however, by applying a trust-region algorithm for optimization, we can find local minimums.

**Theorem 1** *The gradient of $f$ can be computed as*

$$\frac{\partial f}{\partial \lambda_s} = \sum_{i=1}^{n} \left( \dot{\ell}_i(\tilde{u}_{/i}) \times \frac{\partial \tilde{u}_{/i}}{\partial \lambda_s} \right),$$

*where*

$$\begin{aligned}
\frac{\partial \tilde{u}_{/i}}{\partial \lambda_s} &= \frac{\partial \tilde{u}_{/i}}{\partial u_i} \times \frac{\partial u_i}{\partial \lambda_s} + \frac{\partial \tilde{u}_{/i}}{\partial h_i} \times \frac{\partial h_i}{\partial \lambda_s}, \\
\frac{\partial u_i}{\partial \lambda_s} &= \boldsymbol{x}_i^\top \frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s}, \\
\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s} &= -\boldsymbol{H}^{-1} \frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}}), \\
\frac{\partial h_i}{\partial \lambda_s} &= -\boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \frac{\partial \boldsymbol{H}}{\partial \lambda_s} \boldsymbol{H}^{-1} \boldsymbol{x}_i, \text{ and} \\
\frac{\partial \boldsymbol{H}}{\partial \lambda_s} &= \boldsymbol{X}^\top \frac{\partial \boldsymbol{A}}{\partial \lambda_s} \boldsymbol{X} + \frac{\partial \boldsymbol{W}}{\partial \lambda_s}.
\end{aligned}$$

*The diagonal matrices $\boldsymbol{A}$ and $\boldsymbol{W}$ have derivatives*

$$\begin{aligned}
\left( \frac{\partial \boldsymbol{A}}{\partial \lambda_s} \right)_{ii} &= \dddot{\ell}_i(u_i) \times \frac{\partial u_i}{\partial \lambda_s} \quad \text{and} \\
\left( \frac{\partial \boldsymbol{W}}{\partial \lambda_s} \right)_{jj} &= \frac{\partial \ddot{r}_{j,\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\beta}_j) + \dddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j) \times \frac{\partial \hat{\beta}_j}{\partial \lambda_s},
\end{aligned}$$

*and $\tilde{u}_{/i}$ has derivatives*

$$\begin{aligned}
\frac{\partial \tilde{u}_{/i}}{\partial u_i} &= \frac{1}{1 - \ddot{\ell}_i(u_i)\, h_i} + \frac{\dot{\ell}_i(u_i)\, \dddot{\ell}_i(u_i)\, h_i^2}{(1 - \ddot{\ell}_i(u_i)\, h_i)^2} \quad \text{and} \\
\frac{\partial \tilde{u}_{/i}}{\partial h_i} &= \frac{\dot{\ell}_i(u_i)}{(1 - \ddot{\ell}_i(u_i)\, h_i)^2}.
\end{aligned}$$

**Theorem 2** *The hessian of $f$ can be computed as*

$$\frac{\partial^2 f}{\partial \lambda_s \partial \lambda_t} = \sum_{i=1}^{n} \left( \ddot{\ell}_i(\tilde{u}_{/i}) \times \frac{\partial \tilde{u}_{/i}}{\partial \lambda_s} \times \frac{\partial \tilde{u}_{/i}}{\partial \lambda_t} + \dot{\ell}_i(\tilde{u}_{/i}) \times \frac{\partial^2 \tilde{u}_{/i}}{\partial \lambda_s \partial \lambda_t} \right),$$

*where*

$$\begin{aligned}
\frac{\partial^2 \tilde{u}_{/i}}{\partial\lambda_s\partial\lambda_t} &= \frac{\partial \tilde{u}_{/i}}{\partial u_i} \times \frac{\partial^2 u_i}{\partial\lambda_s\partial\lambda_t} + \frac{\partial^2 \tilde{u}_{/i}}{\partial u_i^2} \times \frac{\partial u_i}{\partial\lambda_s} \times \frac{\partial u_i}{\partial\lambda_t} \\
&\quad + \frac{\partial^2 \tilde{u}_{/i}}{\partial u_i \partial h_i} \times \left[\frac{\partial u_i}{\partial\lambda_s} \times \frac{\partial h_i}{\partial\lambda_t} + \frac{\partial u_i}{\partial\lambda_t} \times \frac{\partial h_i}{\partial\lambda_s}\right] \\
&\quad + \frac{\partial \tilde{u}_{/i}}{\partial h_i} \times \frac{\partial^2 h_i}{\partial\lambda_s\partial\lambda_t} + \frac{\partial^2 \tilde{u}_{/i}}{\partial h_i^2} \times \frac{\partial h_i}{\partial\lambda_s} \times \frac{\partial h_i}{\partial\lambda_t}, \\
\frac{\partial^2 u_i}{\partial\lambda_s\partial\lambda_t} &= \boldsymbol{x}_i^\top \frac{\partial^2 \hat{\boldsymbol{\beta}}}{\partial\lambda_s\partial\lambda_t}, \\
\frac{\partial^2 \hat{\boldsymbol{\beta}}}{\partial\lambda_s\partial\lambda_t} &= -\boldsymbol{H}^{-1}\boldsymbol{X}^\top \cdot \operatorname{vec}\left[\left\{\dddot{\ell}_i(u_i) \times \frac{\partial u_i}{\partial\lambda_s} \times \frac{\partial u_i}{\partial\lambda_t}\right\}_i\right] \\
&\quad - \boldsymbol{H}^{-1}\frac{\partial\nabla^2 R_{\boldsymbol{\lambda}}}{\partial\lambda_s}(\hat{\boldsymbol{\beta}})\frac{\partial\hat{\boldsymbol{\beta}}}{\partial\lambda_t} - \boldsymbol{H}^{-1}\frac{\partial\nabla^2 R_{\boldsymbol{\lambda}}}{\partial\lambda_t}(\hat{\boldsymbol{\beta}})\frac{\partial\hat{\boldsymbol{\beta}}}{\partial\lambda_s} - \boldsymbol{H}^{-1}\frac{\partial\nabla R_{\boldsymbol{\lambda}}}{\partial\lambda_s\partial\lambda_t}(\hat{\boldsymbol{\beta}}) \\
&\quad - \boldsymbol{H}^{-1} \cdot \operatorname{vec}\left[\left\{\dddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j) \times \frac{\partial\hat{\beta}_j}{\partial\lambda_s} \times \frac{\partial\hat{\beta}_j}{\partial\lambda_t}\right\}_j\right], \\
\frac{\partial^2 h_i}{\partial\lambda_s\partial\lambda_t} &= 2\boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial\boldsymbol{H}}{\partial\lambda_s}\boldsymbol{H}^{-1}\frac{\partial\boldsymbol{H}}{\partial\lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial^2\boldsymbol{H}}{\partial\lambda_s\partial\lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i, \text{ and} \\
\frac{\partial^2 \boldsymbol{H}}{\partial\lambda_s\partial\lambda_t} &= \boldsymbol{X}^\top \frac{\partial^2\boldsymbol{A}}{\partial\lambda_s\partial\lambda_t}\boldsymbol{X} + \frac{\partial^2\boldsymbol{W}}{\partial\lambda_s\partial\lambda_t}.
\end{aligned}$$

*The diagonal matrices $\boldsymbol{A}$ and $\boldsymbol{W}$ have second derivatives*

$$\begin{aligned}
\left(\frac{\partial^2\boldsymbol{A}}{\partial\lambda_s\partial\lambda_t}\right)_{ii} &= \dddot{\ell}_i(u_i) \times \frac{\partial^2 u_i}{\partial\lambda_s\partial\lambda_t} + \ddddot{\ell}_i(u_i) \times \frac{\partial u_i}{\partial\lambda_s} \times \frac{\partial u_i}{\partial\lambda_t} \quad \text{and} \\
\left(\frac{\partial^2\boldsymbol{W}}{\partial\lambda_s\partial\lambda_t}\right)_{jj} &= \frac{\partial^2\ddot{r}_{j,\boldsymbol{\lambda}}}{\partial\lambda_s\partial\lambda_t}(\hat{\beta}_j) + \frac{\partial\dddot{r}_{j,\boldsymbol{\lambda}}}{\partial\lambda_s}(\hat{\beta}_j) \times \frac{\partial\hat{\beta}_j}{\partial\lambda_t} + \frac{\partial\dddot{r}_{j,\boldsymbol{\lambda}}}{\partial\lambda_t}(\hat{\beta}_j) \times \frac{\partial\hat{\beta}_j}{\partial\lambda_s} \\
&\quad + \dddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j) \times \frac{\partial^2\hat{\beta}_j}{\partial\lambda_s\partial\lambda_t} + \ddddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j) \times \frac{\partial\hat{\beta}_j}{\partial\lambda_s} \times \frac{\partial\hat{\beta}_j}{\partial\lambda_t},
\end{aligned}$$

*and $\tilde{u}_{/i}$ has second derivatives*

$$\begin{aligned}
\frac{\partial^2\tilde{u}_{/i}}{\partial u_i^2} &= \frac{\dddot{\ell}_i(u_i)\,h_i}{(1-\ddot{\ell}_i(u_i)\,h_i)^2} + \frac{(\ddot{\ell}_i(u_i)\,\dddot{\ell}_i(u_i) + \dot{\ell}_i(u_i)\,\ddddot{\ell}_i(u_i))h_i^2}{(1-\ddot{\ell}_i(u_i)\,h_i)^2} + \frac{2\dot{\ell}_i(u_i)\,\dddot{\ell}_i(u_i)^2\,h_i^3}{(1-\ddot{\ell}_i(u_i)\,h_i)^3}, \\
\frac{\partial^2\tilde{u}_{/i}}{\partial u_i\partial h_i} &= \frac{\ddot{\ell}_i(u_i)}{(1-\ddot{\ell}_i(u_i)\,h_i)^2} + \frac{2\dot{\ell}_i(u_i)\,\dddot{\ell}_i(u_i)\,h_i}{(1-\ddot{\ell}_i(u_i)\,h_i)^3}, \text{ and} \\
\frac{\partial^2\tilde{u}_{/i}}{\partial h_i^2} &= \frac{2\dot{\ell}_i(u_i)\,\ddot{\ell}_i(u_i)}{(1-\ddot{\ell}_i(u_i)\,h_i)^3}.
\end{aligned}$$

### 4.1 Computational Complexity

Let $q$ denote the number of hyperparameters. How best to proceed with the computations will depend on which is greater $n$ or $p$. We assume first that $n > p$. $\boldsymbol{H}$ and its Cholesky

factorization $\boldsymbol{L}$ can be computed in $\mathcal{O}\left(p^2 n\right)$ operations. Let $\boldsymbol{h}$ denote the vector of $h_i$ values. The complexity of computing the ALO value, gradient, and hessian is dominated by the cost of evaluating $\boldsymbol{h}$ and its derivatives. Using the formula

$$h_i = \|\boldsymbol{L}^{-1}\boldsymbol{x}_i\|^2,$$

$\boldsymbol{h}$ can be computed with $\mathcal{O}\left(p^2 n\right)$ operations. For the derivatives of $\boldsymbol{h}$, we first compute the matrices $\frac{\partial \boldsymbol{H}}{\partial \lambda_s}$, which can be done in $\mathcal{O}\left(p^2 q n\right)$ operations. The derivatives can then, also, be computed with $\mathcal{O}\left(p^2 q n\right)$ operations using the formulas

$$\boldsymbol{t}_i = {\boldsymbol{L}^{-1}}^\top \boldsymbol{L}^{-1}\boldsymbol{x}_i \quad \text{and} \quad \frac{\partial h_i}{\partial \lambda_s} = \boldsymbol{t}_i^\top \frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{t}_i.$$

For the second derivatives of $\boldsymbol{h}$, we, similarly, first compute the matrices $\frac{\partial^2 \boldsymbol{H}}{\partial \lambda_s \partial \lambda_t}$, which can be done in $\mathcal{O}\left(p^2 q^2 n\right)$ operations, and then compute

$$\boldsymbol{r}_{si} = \boldsymbol{L}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{H}^{-1}\boldsymbol{x}_i \quad \text{and} \quad \frac{\partial^2 h_i}{\partial \lambda_s \partial \lambda_t} = \boldsymbol{r}_{si}^\top \boldsymbol{r}_{ti} - \boldsymbol{t}_i^\top \frac{\partial^2 \boldsymbol{H}}{\partial \lambda_s \partial \lambda_t}\boldsymbol{t}_i$$

with $\mathcal{O}\left(p^2 q^2 n\right)$ operations.

When $p > n$ and $\boldsymbol{W}$ is nonsingular, we can achieve better complexity by evaluating the equations in a different order. We apply the matrix inversion lemma to $\boldsymbol{H}^{-1} = \left[\boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X} + \boldsymbol{W}\right]^{-1}$ to obtain

$$\boldsymbol{H}^{-1} = \boldsymbol{W}^{-1} + \boldsymbol{W}^{-1}\boldsymbol{X}^\top \left[\boldsymbol{A}^{-1} + \boldsymbol{X}\boldsymbol{W}^{-1}\boldsymbol{X}^\top\right]^{-1} \boldsymbol{X}\boldsymbol{W}^{-1}.$$

Computing the matrix $\boldsymbol{A}^{-1} + \boldsymbol{X}\boldsymbol{W}^{-1}\boldsymbol{X}^\top$ and its Cholesky factorization can be done in $\mathcal{O}\left(n^2 p\right)$ operations, and a product $\boldsymbol{H}^{-1}b$ can be done in $\mathcal{O}\left(np\right)$ operations. Applying the same approach, where we avoid explicitly computing the $p$-by-$p$ matrices, we can compute the gradient and hessian in $\mathcal{O}\left(n^2 q p\right)$ and $\mathcal{O}\left(n^2 q^2 p\right)$ operations, respectively. If $\boldsymbol{W}$ is singular but only has $k$ zero diagonal entries, where $k \ll p$, we can combine this approach with block matrix inversion and still achieve more efficient formulas.

## 5. Examples

By customizing $\ell_i$ and $R_\lambda$, we can adopt Theorem 1 and Theorem 2 to a broad range of models. Putting $\ell_i(u) \triangleq (y_i - u)^2$ gives us the regularized least squares; putting $\ell_i(u) \triangleq \log\left[1 + \exp(-y_i u)\right]$ gives us regularized logistic regression. If we define

$$R_\lambda(\boldsymbol{\beta}) \triangleq \lambda^2 \left\|\boldsymbol{\beta}\right\|^2,$$

we have standard ridge regularization. Defining

$$R_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) \triangleq \sum_{j=1}^{p} \lambda_{g_j}^2 |\beta_j|^2$$

allows us to use different regularization strengths for different groups of variables, and putting

$$R_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) \triangleq \lambda_1^2 \sum_{j=1}^{p} |\beta_j|^{1+\lambda_2^2}$$

gives us bridge regularization (Fu , 1998). We present versions of Theorem 1 and Theorem 2 for the specific cases of ridge regression and logistic regression with ridge regularization.

### 5.1 Ridge Regression

Because Equation 1 is a quadratic for ridge regression, the Newton approximation step in ALO is exact, $\tilde{\boldsymbol{\beta}}_{/i} = \hat{\boldsymbol{\beta}}_{/i}$, and ALO and LO are equivalent. Before stating theorems for the LO gradient and hessian, we first introduce more familiar notation. Put

$$\hat{y}_i \triangleq u_i, \quad \hat{y}_{/i} \triangleq \tilde{u}_{/i}, \quad \varepsilon_i \triangleq y_i - \hat{y}_i, \quad \text{and } \varepsilon_{/i} \triangleq y_i - \hat{y}_{/i}.$$

$\hat{y}_i$ and $\hat{y}_{/i}$ denote the $i^{\text{th}}$ in-sample and out-of-sample prediction, respectively; and $\varepsilon_i$ and $\varepsilon_{/i}$ denote the $i^{\text{th}}$ in-sample and out-of-sample prediction error, respectively. Using this notation,

$$\text{LO}_{\boldsymbol{\lambda}} = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{/i}^2,$$

where

$$\varepsilon_{/i} = \frac{\varepsilon_i}{1 - 2h_i} \quad \text{and} \quad h_i = \frac{1}{2} \boldsymbol{x}_i^\top \left( \boldsymbol{X}^\top \boldsymbol{X} + \lambda^2 \boldsymbol{I} \right)^{-1} \boldsymbol{x}_i.$$

**Corollary 3** *For ridge regression, the gradient of $f$ can be computed as*

$$\frac{\partial f}{\partial \lambda} = \sum_{i=1}^{n} \left( -2\varepsilon_{/i} \times \frac{\partial \hat{y}_{/i}}{\partial \lambda} \right),$$

*where*

$$\begin{aligned}
\frac{\partial \hat{y}_{/i}}{\partial \lambda} &= \frac{\partial \hat{y}_{/i}}{\partial \hat{y}_i} \times \frac{\partial \hat{y}_i}{\partial \lambda} + \frac{\partial \hat{y}_{/i}}{\partial h_i} \times \frac{\partial h_i}{\partial \lambda}, \\
\frac{\partial \hat{y}_i}{\partial \lambda} &= -2\lambda \boldsymbol{x}_i^\top \left( \boldsymbol{X}^\top \boldsymbol{X} + \lambda^2 \boldsymbol{I} \right)^{-1} \hat{\boldsymbol{\beta}}, \text{ and} \\
\frac{\partial h_i}{\partial \lambda} &= -\lambda \boldsymbol{x}_i^\top \left( \boldsymbol{X}^\top \boldsymbol{X} + \lambda^2 \boldsymbol{I} \right)^{-2} \boldsymbol{x}_i,
\end{aligned}$$

*and $\hat{y}_{/i}$ has derivatives*

$$\frac{\partial \hat{y}_{/i}}{\partial \hat{y}_i} = \frac{1}{1 - 2h_i} \quad \text{and} \quad \frac{\partial \hat{y}_{/i}}{\partial h_i} = \frac{-2\varepsilon_i}{(1 - 2h_i)^2}.$$

**Corollary 4** *For ridge regression, the hessian of $f$ can be computed as*

$$\frac{\partial^2 f}{\partial \lambda^2} = \sum_{i=1}^{n} \left[ 2 \left( \frac{\partial \hat{y}_{/i}}{\partial \lambda} \right)^2 - 2\varepsilon_{/i} \times \frac{\partial^2 \hat{y}_{/i}}{\partial \lambda^2} \right],$$

*where*

$$\begin{aligned}
\frac{\partial^2 \hat{y}_{/i}}{\partial \lambda^2} &= \frac{\partial \hat{y}_{/i}}{\partial \hat{y}_i} \times \frac{\partial^2 \hat{y}_i}{\partial \lambda^2} + 2 \frac{\partial^2 \hat{y}_{/i}}{\partial \hat{y}_i \partial h_i} \times \frac{\partial \hat{y}_i}{\partial \lambda} \times \frac{\partial h_i}{\partial \lambda} \\
&\quad + \frac{\partial \hat{y}_{/i}}{\partial h_i} \times \frac{\partial^2 h_i}{\partial \lambda^2} + \frac{\partial^2 \hat{y}_{/i}}{\partial h_i^2} \times \left( \frac{\partial h_i}{\partial \lambda} \right)^2, \\
\frac{\partial^2 \hat{y}_i}{\partial \lambda^2} &= 8\lambda^2 \boldsymbol{x}_i^\top \left( \boldsymbol{X}^\top \boldsymbol{X} + \lambda^2 \boldsymbol{I} \right)^{-2} \hat{\boldsymbol{\beta}} - 2\boldsymbol{x}^\top \left( \boldsymbol{X}^\top \boldsymbol{X} + \lambda^2 \boldsymbol{I} \right)^{-1} \hat{\boldsymbol{\beta}}, \text{ and} \\
\frac{\partial^2 h_i}{\partial \lambda^2} &= 4\lambda^2 \boldsymbol{x}_i^\top \left( \boldsymbol{X}^\top \boldsymbol{X} + \lambda^2 \boldsymbol{I} \right)^{-3} \boldsymbol{x}_i - \boldsymbol{x}_i^\top \left( \boldsymbol{X}^\top \boldsymbol{X} + \lambda^2 \boldsymbol{I} \right)^{-2} \boldsymbol{x}_i,
\end{aligned}$$

*and $\hat{y}_{/i}$ has second derivatives*

$$\frac{\partial^2 \hat{y}_{/i}}{\partial \hat{y}_i \partial h_i} = \frac{2}{(1-2h_i)^2} \quad \text{and} \quad \frac{\partial^2 \hat{y}_{/i}}{\partial h_i^2} = \frac{-8\varepsilon_i}{(1-2h_i)^3}.$$

### 5.2 Ridge Regularized Logistic Regression

For ridge regularized logistic regression, we have

$$\begin{aligned}
\boldsymbol{H} &= \boldsymbol{X}^\top \boldsymbol{A} \boldsymbol{X} + 2\lambda^2 \boldsymbol{I}, \\
\ell_i(u) &\triangleq \log\left[1 + \exp(-y_i u)\right], \\
\dot{\ell}_i\left(u_i\right) &= \frac{-y_i}{1 + \exp(y_i u)}, \\
\ddot{\ell}_i\left(u_i\right) &= qp, \\
\dddot{\ell}_i\left(u\right) &= qp\left(q - p\right), \text{ and} \\
\ddddot{\ell}_i\left(u\right) &= qp\left(q^2 + p^2\right) - 4q^2p^2,
\end{aligned}$$

with $p = (1 + \exp(-u))^{-1}$ and $q = 1 - p$.

**Corollary 5** *For ridge regularized logistic regression, the gradient of $f$ can be computed as*

$$\frac{\partial f}{\partial \lambda} = \sum_{i=1}^{n} \left( \dot{\ell}_i(\tilde{u}_{/i}) \times \frac{\partial \tilde{u}_{/i}}{\partial \lambda} \right),$$

*where*

$$\begin{aligned}
\frac{\partial \tilde{u}_{/i}}{\partial \lambda} &= \frac{\partial \tilde{u}_{/i}}{\partial u_i} \times \frac{\partial u_i}{\partial \lambda} + \frac{\partial \tilde{u}_{/i}}{\partial h_i} \times \frac{\partial h_i}{\partial \lambda}, \\
\frac{\partial u_i}{\partial \lambda} &= -4\lambda \boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \hat{\boldsymbol{\beta}}, \\
\frac{\partial h_i}{\partial \lambda} &= -\boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \frac{\partial \boldsymbol{H}}{\partial \lambda} \boldsymbol{H}^{-1} \boldsymbol{x}_i, \text{ and} \\
\frac{\partial \boldsymbol{H}}{\partial \lambda} &= \boldsymbol{X}^\top \frac{\partial \boldsymbol{A}}{\partial \lambda} \boldsymbol{X} + 4\lambda \boldsymbol{I}.
\end{aligned}$$

*The diagonal matrix* $\boldsymbol{A}$ *has derivative*

$$\left(\frac{\partial \boldsymbol{A}}{\partial \lambda}\right)_{ii} = \dddot{\ell}_i(u_i) \times \frac{\partial u_i}{\partial \lambda},$$

*and* $\tilde{u}_{/i}$ *has derivatives*

$$\begin{aligned}
\frac{\partial \tilde{u}_{/i}}{\partial u_i} &= \frac{1}{1 - \ddot{\ell}_i(u_i)\, h_i} + \frac{\dot{\ell}_i(u_i)\, \dddot{\ell}_i(u_i)\, h_i^2}{(1 - \ddot{\ell}_i(u_i)\, h_i)^2} \quad \textit{and} \\
\frac{\partial \tilde{u}_{/i}}{\partial h_i} &= \frac{\dot{\ell}_i(u_i)}{(1 - \ddot{\ell}_i(u_i)\, h_i)^2}.
\end{aligned}$$

**Corollary 6** *For ridge regularized logistic regression, the hessian of* $f$ *can be computed as*

$$\frac{\partial^2 f}{\partial \lambda^2} = \sum_{i=1}^{n} \left[ \ddot{\ell}_i(\tilde{u}_{/i}) \times \left(\frac{\partial \tilde{u}_{/i}}{\partial \lambda}\right)^2 + \dot{\ell}_i(\tilde{u}_{/i}) \times \frac{\partial^2 \tilde{u}_{/i}}{\partial \lambda^2} \right],$$

*where*

$$\begin{aligned}
\frac{\partial^2 \tilde{u}_{/i}}{\partial \lambda^2} &= \frac{\partial \tilde{u}_{/i}}{\partial u_i} \times \frac{\partial^2 u_i}{\partial \lambda^2} + \frac{\partial^2 \tilde{u}_{/i}}{\partial u_i^2} \times \left(\frac{\partial u_i}{\partial \lambda}\right)^2 \\
&\quad + 2\frac{\partial^2 \tilde{u}_{/i}}{\partial u_i \partial h_i} \times \frac{\partial u_i}{\partial \lambda} \times \frac{\partial h_i}{\partial \lambda} \\
&\quad + \frac{\partial \tilde{u}_{/i}}{\partial h_i} \times \frac{\partial^2 h_i}{\partial \lambda^2} + \frac{\partial^2 \tilde{u}_{/i}}{\partial h_i^2} \times \left(\frac{\partial h_i}{\partial \lambda}\right)^2, \\
\frac{\partial^2 u_i}{\partial \lambda^2} &= -\boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \boldsymbol{X}^\top \cdot \operatorname{vec}\left[ \left\{ \dddot{\ell}_i(u_i) \times \left(\frac{\partial u_i}{\partial \lambda}\right)^2 \right\}_i \right] \\
&\quad + 32\lambda^2 \boldsymbol{x}_i^\top \boldsymbol{H}^{-2} \hat{\boldsymbol{\beta}} - 4\boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \hat{\boldsymbol{\beta}}, \\
\frac{\partial^2 h_i}{\partial \lambda^2} &= 2\boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \frac{\partial \boldsymbol{H}}{\partial \lambda} \boldsymbol{H}^{-1} \frac{\partial \boldsymbol{H}}{\partial \lambda} \boldsymbol{H}^{-1} \boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \frac{\partial^2 \boldsymbol{H}}{\partial \lambda^2} \boldsymbol{H}^{-1} \boldsymbol{x}_i, \ \textit{and} \\
\frac{\partial^2 \boldsymbol{H}}{\partial \lambda^2} &= \boldsymbol{X}^\top \frac{\partial^2 \boldsymbol{A}}{\partial \lambda^2} \boldsymbol{X} + 4\boldsymbol{I}.
\end{aligned}$$

*The diagonal matrix* $\boldsymbol{A}$ *has second derivative*

$$\left(\frac{\partial^2 \boldsymbol{A}}{\partial \lambda^2}\right)_{ii} = \dddot{\ell}_i(u_i) \times \frac{\partial^2 u_i}{\partial \lambda^2} + \ddddot{\ell}_i(u_i) \times \left(\frac{\partial u_i}{\partial \lambda}\right)^2,$$

*and* $\tilde{u}_{/i}$ *has second derivatives*

$$\begin{aligned}
\frac{\partial^2 \tilde{u}_{/i}}{\partial u_i^2} &= \frac{\dddot{\ell}_i(u_i)\, h_i}{(1 - \ddot{\ell}_i(u_i)\, h_i)^2} + \frac{(\ddot{\ell}_i(u_i)\, \dddot{\ell}_i(u_i) + \dot{\ell}_i(u_i)\, \ddddot{\ell}_i(u_i)) h_i^2}{(1 - \ddot{\ell}_i(u_i)\, h_i)^2} + \frac{2\dot{\ell}_i(u_i)\, \dddot{\ell}_i(u_i)^2\, h_i^3}{(1 - \ddot{\ell}_i(u_i)\, h_i)^3}, \\
\frac{\partial^2 \tilde{u}_{/i}}{\partial u_i \partial h_i} &= \frac{\ddot{\ell}_i(u_i)}{(1 - \ddot{\ell}_i(u_i)\, h_i)^2} + \frac{2\dot{\ell}_i(u_i)\, \dddot{\ell}_i(u_i)\, h_i}{(1 - \ddot{\ell}_i(u_i)\, h_i)^3}, \ \textit{and} \\
\frac{\partial^2 \tilde{u}_{/i}}{\partial h_i^2} &= \frac{2\dot{\ell}_i(u_i)\, \ddot{\ell}_i(u_i)}{(1 - \ddot{\ell}_i(u_i)\, h_i)^3}.
\end{aligned}$$

## 6. Numerical Experiments

We run experiments designed around these lines of inquiry:

1. What do the derivatives of ALO look like?
2. How do hyperparameters found by ALO optimization compare to those found by grid search?
3. What is the cost of ALO optimization?
4. Can we use ALO optimization to fit models with multiple hyperparameters that lead to better performance on out-of-sample predictions?

To that end, we fit ridge regression and regularized logistic regression models to real-world sample data sets. Table 1 catalogs the data sets used, and we provide brief summaries below.

| **Data Set** | **Task** | **n** | **p** |
|---|---|---|---|
| `Breast Cancer` | classification | 569 | 30 |
| `Cleveland Heart` | classification | 297 | 22 |
| `Pollution` | regression | 60 | 15 |
| `Arcene` | classification | 200 | 10000 |
| `Gisette` | classification | 7000 | 5000 |

Table 1: Sample data sets used in experiments

`Breast Cancer` is a binary classification data set where the objective is to predict whether breast mass is malignant from characteristics of cell nuclei.

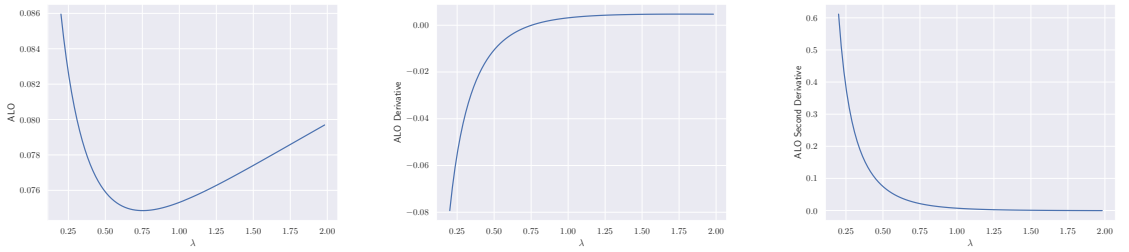
`Cleveland Heart` is a binary classification data set where the objective is to detect the presence of heart disease. The data set uses 13 features, but 5 are categorical. We obtain 22 features after transforming the categorical features to indicator variables, and we obtain 297 entries after dropping any entries with missing features.

`Pollution` is a regression data set where the objective is to predict the mortality rate of metropolitan areas from environmental and socioeconomic variables (McDonald and Schwing, 1973).

`Arcene` is a binary classification data set where the objective is to distinguish cancer versus normal patterns from mass-spectrometric data. We combine the `Arcene` training and validation data sets to get a data set with 200 entries.

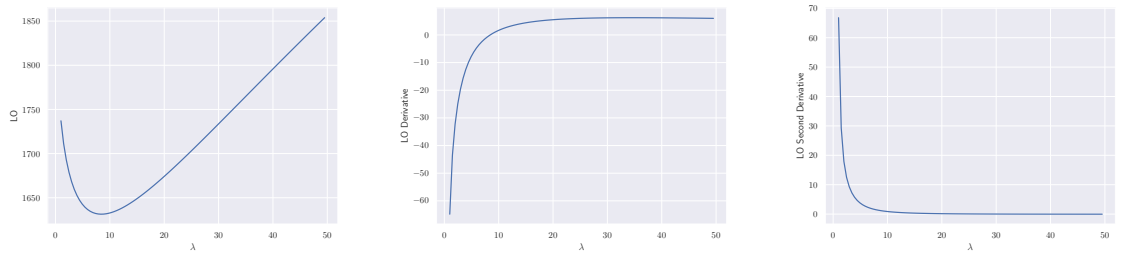
`Gisette` is a binary classification data set where the objective is to predict whether a handwritten digit is a 4 or a 9. The data set includes features derived from a 28x28 pixel digit image and non-predictive probe features. For `Gisette`, we again combine the training and validation data sets to get 7000 entries.

We preprocess all data sets used for training so that features have zero mean and unit standard deviation when not constant.




(a) `ALO` (b) `ALO Derivative` (c) `ALO Second Derivative`

Figure 1: ALO for regularized logistic regression on the `Breast Cancer` data set




(a) `LO` (b) `LO Derivative` (c) `LO Second Derivative`

Figure 2: LO for ridge regression on the `Pollution` data set

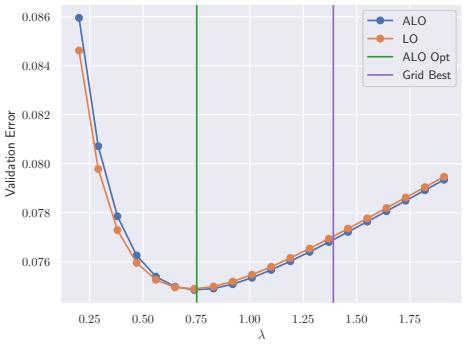
### 6.1 Visualizing ALO and its Derivatives

We begin by graphing ALO and its derivatives on sample data sets. Figure 1 plots ALO and its derivatives for logistic regression with the ridge regularizer $R_\lambda(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|^2$ on the `Breast Cancer` data set. Figure 2 plots LO and its derivatives for ridge regression on the `Pollution` data set. With ridge regression, ALO and LO are equivalent.

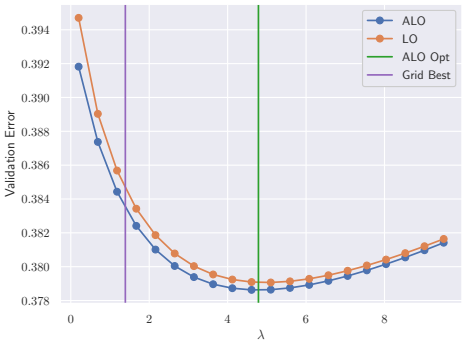
### 6.2 Comparing ALO Optimization to Grid Search

We next compare hyperparameters found by ALO optimization to those found by grid search for the ridge regularizer. For ALO optimization, we use the Python package `peak-engines` available from `https://github.com/rnburn/peak-engines`; and for grid search, we use `LogisticRegressionCV` and `RidgeCV` from `sklearn-0.23.1` with default settings. `LogisticRegressionCV` defaults to run a grid search using 5-fold cross-validation and 10 points of evaluation; `RidgeCV` defaults to run a grid search using LO and 3 points of evaluation.[1] Figure 3 shows where the hyperparameters found by grid search and ALO optimization lie along the the LO and ALO curves. We additionally benchmark how long it takes ALO optimization and grid search to find hyperparameters on the sample data sets. Figure 4 shows the resulting durations measured in seconds.

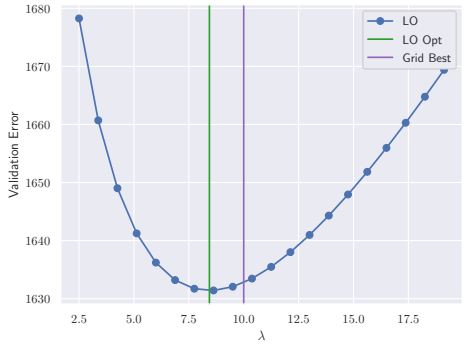
1. `RidgeCV`'s documentation claims that it uses Generalized Cross-Validation, but it's actually using LO. See `https://github.com/scikit-learn/scikit-learn/issues/18079`.



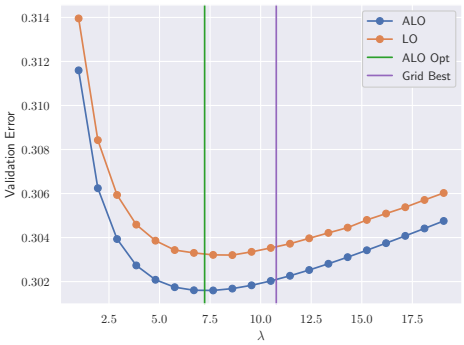

(a) `Breast Cancer`




(b) `Cleveland Heart`




(c) `Pollution`




(d) `Arcene`

Figure 3: Plot of ALO and LO curves for sample data sets. The green line shows the hyperparameter found by ALO optimization, and the purple line shows the hyperparameter found by grid search.

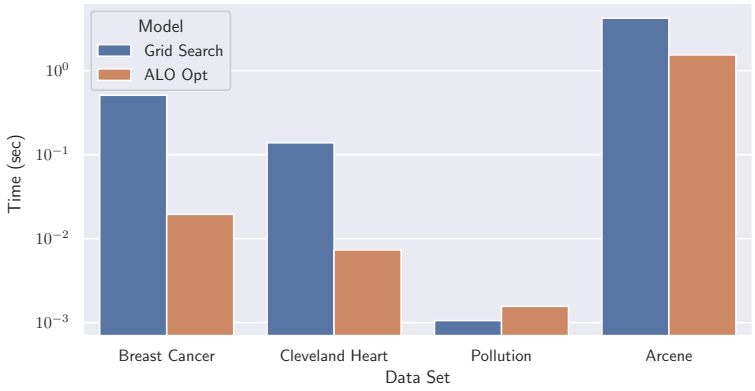



Figure 4: Benchmark showing how long ALO optimization and grid search take to find hyperparameters on sample data sets. The times are averaged over 10 runs and shown on a logarithmic scale.

### 6.3 Fitting Models with Multiple Hyperparameters

To test ALO optimization with multiple hyperparameters, we fit logistic regression with a bridge regularizer (Fu , 1998) of the form $\lambda_1|\beta_j|^{\lambda_2}$, where $\lambda_2 \geq 1$, to a 5-fold cross-validation of the `Gisette` data set. To ensure that $r_{j,\boldsymbol{\lambda}}$ has a continuous fourth derivative, we interpolate the regularizer with a polynomial when $|\beta_j| < \delta$ so that

$$r_{j,\boldsymbol{\lambda}}(\beta_j) = \begin{cases} \lambda_1|\beta_j|^{\lambda_2}, & \text{if } |\beta_j| \geq \delta, \\ \lambda_1 p_{\lambda_2}(|\beta_j|), & \text{if } |\beta_j| < \delta, \end{cases}$$

where $p_{\lambda_2}(t) = a_1 t^2 + a_2 t^4 + a_3 t^5 + a_4 t^6 + a_5 t^7$ and $\boldsymbol{a}$ is chosen so that $p_{\lambda_2}(\delta)$ matches $\delta^{\lambda_2}$ up to the fourth derivative. For our experiment, we use $\delta = 0.01$. For comparison, we also fit logistic regression with ridge regularization. Table 2 shows the hyperparameters found for bridge regularization and ridge regularization and compares their performance on each cross-validation fold.

## 7. Conclusion

In this paper, we demonstrated how to select hyperparameters by computing the gradient and hessian of ALO and applying a second-order optimizer to find a local minimum. The approach is applicable to a large class of commonly used models, including regularized logistic regression and ridge regression. We applied ALO optimization to fit regularized models to various real-world data sets. We found that when using a single-parameter regularizer, we were able to find hyperparameters with better LO values than standard grid-search approaches and frequently were able to do so in less time. ALO optimization, furthermore, scales to handle multiple hyperparameters, and we demonstrated how it could be used to fit hyperparameters for bridge regularization.

| Fold | $\boldsymbol{\lambda}_1$ | $\boldsymbol{\lambda}_2$ | Test Error |
|---|---|---|---|
| 0 | 17.3 | 1.95 | 0.0698 |
| 1 | 16.8 | 1.94 | 0.0581 |
| 2 | 13.0 | 1.81 | 0.0667 |
| 3 | 10.5 | 1.75 | 0.0677 |
| 4 | 10.8 | 1.83 | 0.0639 |
| **Mean** | | | 0.0652 |

(a) Bridge Regularization

| Fold | $\boldsymbol{\lambda}$ | Test Error |
|---|---|---|
| 0 | 19.1 | 0.0709 |
| 1 | 18.7 | 0.0587 |
| 2 | 18.3 | 0.0697 |
| 3 | 16.5 | 0.0714 |
| 4 | 14.4 | 0.0667 |
| **Mean** | | 0.0675 |

(b) Ridge Regularization

Table 2: Hyperparameters and performance of ALO optimization with logistic regression using bridge and ridge regularization on a 5-fold cross-validation of the `Gisette` data set. Test error is measured as the negative mean log-likelihood of the out-of-sample fold.

## Acknowledgments

We made use of the UCI Machine Learning Repository `http://archive.ics.uci.edu/ml` in our experiments.

## Appendix A. Proof of Theorem 1

To derive the derivative of $\hat{\boldsymbol{\beta}}$, we observe that at the optimum of Equation 1 the gradient is zero:

$$\sum_{i=1}^{n} \dot{\ell}_i\left(u_i\right) \boldsymbol{x}_i + \nabla R_{\boldsymbol{\lambda}}(\hat{\boldsymbol{\beta}}) = 0.$$

Differentiating both sides of the equation gives us

$$\begin{aligned}
&\frac{\partial}{\partial \lambda_s}\left(\sum_{i=1}^{n} \dot{\ell}_i\left(u_i\right) \boldsymbol{x}_i + \nabla R_{\boldsymbol{\lambda}}(\hat{\boldsymbol{\beta}})\right) = 0 \\
\Leftrightarrow\ &\boldsymbol{H}\left(\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s}\right) + \frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}}) = 0 \\
\Leftrightarrow\ &\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s} = -\boldsymbol{H}^{-1}\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}}).
\end{aligned}$$

For the derivative of $h_i$, we apply this formula for differentiating an inverse matrix:

$$\frac{d\boldsymbol{E}^{-1}}{dt} = -\boldsymbol{E}^{-1}\frac{d\boldsymbol{E}}{dt}\boldsymbol{E}^{-1}.$$

The other derivatives are derived as straightforward differentiations of their associated value formulas.

## Appendix B. Proof of Theorem 2

For the second derivative of $\hat{\boldsymbol{\beta}}$, we derive

$$\begin{aligned}
\frac{\partial^2 \hat{\boldsymbol{\beta}}}{\partial \lambda_s \partial \lambda_t} &= \frac{\partial}{\partial \lambda_t}\left[-\boldsymbol{H}^{-1}\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\right] \\
&= \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_t}\boldsymbol{H}^{-1}\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}}) - \boldsymbol{H}^{-1}\frac{\partial}{\partial \lambda_t}\left(\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\right) \\
&= -\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_t}\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s} - \boldsymbol{H}^{-1}\frac{\partial}{\partial \lambda_t}\left(\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\right).
\end{aligned}$$

Now,

$$\begin{aligned}
\left(\frac{\partial \boldsymbol{H}}{\partial \lambda_t}\right)\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s} &= \left[\boldsymbol{X}^\top \frac{\partial \boldsymbol{A}}{\partial \lambda_t}\boldsymbol{X} + \frac{\partial \boldsymbol{W}}{\partial \lambda_t}\right]\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s} \\
&= \boldsymbol{X}^\top \cdot \operatorname{vec}\left[\left\{\dddot{\ell}_i(u_i) \times \frac{\partial u_i}{\partial \lambda_s} \times \frac{\partial u_i}{\partial \lambda_t}\right\}_i\right] + \frac{\partial \nabla^2 R_{\boldsymbol{\lambda}}}{\partial \lambda_t}(\hat{\boldsymbol{\beta}})\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s} \\
&\quad + \operatorname{vec}\left[\left\{\dddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j) \times \frac{\partial \hat{\beta}_j}{\partial \lambda_s} \times \frac{\partial \hat{\beta}_j}{\partial \lambda_t}\right\}_j\right] \quad \text{and} \\
\frac{\partial}{\partial \lambda_t}\left(\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\right) &= \frac{\partial \nabla^2 R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_t} + \frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s \partial \lambda_t}(\hat{\boldsymbol{\beta}}).
\end{aligned}$$

Combining the equations, the second derivative of $\hat{\boldsymbol{\beta}}$ becomes

$$\begin{aligned}
\frac{\partial^2 \hat{\boldsymbol{\beta}}}{\partial \lambda_s \partial \lambda_t} &= -\boldsymbol{H}^{-1}\boldsymbol{X}^\top \cdot \operatorname{vec}\left[\left\{\dddot{\ell}_i(u_i) \times \frac{\partial u_i}{\partial \lambda_s} \times \frac{\partial u_i}{\partial \lambda_t}\right\}_i\right] \\
&\quad - \boldsymbol{H}^{-1}\frac{\partial \nabla^2 R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_t} - \boldsymbol{H}^{-1}\frac{\partial \nabla^2 R_{\boldsymbol{\lambda}}}{\partial \lambda_t}(\hat{\boldsymbol{\beta}})\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s} - \boldsymbol{H}^{-1}\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s \partial \lambda_t}(\hat{\boldsymbol{\beta}}) \\
&\quad - \boldsymbol{H}^{-1} \cdot \operatorname{vec}\left[\left\{\dddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j) \times \frac{\partial \hat{\beta}_j}{\partial \lambda_s} \times \frac{\partial \hat{\beta}_j}{\partial \lambda_t}\right\}_j\right].
\end{aligned}$$

For the second derivative of $h_i$, we derive

$$\begin{aligned}
\frac{\partial^2 h_i}{\partial\lambda_s\partial\lambda_t} &= \frac{\partial}{\partial\lambda_t}\left(-\boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial\lambda_s}\boldsymbol{H}^{-1}\boldsymbol{x}_i\right) \\
&= \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial\lambda_t}\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial\lambda_s}\boldsymbol{H}^{-1}\boldsymbol{x}_i + \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial\lambda_s}\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial\lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i \\
&\quad - \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial^2 \boldsymbol{H}}{\partial\lambda_s\partial\lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i \\
&= 2\boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial\lambda_s}\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial\lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial^2 \boldsymbol{H}}{\partial\lambda_s\partial\lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i, \\
\frac{\partial^2 \boldsymbol{H}}{\partial\lambda_s\partial\lambda_t} &= \frac{\partial}{\partial\lambda_t}\left(\boldsymbol{X}^\top\frac{\partial \boldsymbol{A}}{\partial\lambda_s}\boldsymbol{X} + \frac{\partial \boldsymbol{W}}{\partial\lambda_s}\right) \\
&= \boldsymbol{X}^\top\frac{\partial^2 \boldsymbol{A}}{\partial\lambda_s\partial\lambda_t}\boldsymbol{X} + \frac{\partial^2 \boldsymbol{W}}{\partial\lambda_s\partial\lambda_t}, \text{ and} \\
\left(\frac{\partial^2 \boldsymbol{W}}{\partial\lambda_s\partial\lambda_t}\right)_{jj} &= \frac{\partial}{\partial\lambda_t}\left(\frac{\partial \ddot{r}_{j,\boldsymbol{\lambda}}}{\partial\lambda_s}(\hat{\beta}_j) + \dddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j)\times\frac{\partial\hat{\beta}_j}{\partial\lambda_s}\right) \\
&= \frac{\partial^2 \ddot{r}_{j,\boldsymbol{\lambda}}}{\partial\lambda_s\partial\lambda_t}(\hat{\beta}_j) + \frac{\partial \dddot{r}_{j,\boldsymbol{\lambda}}}{\partial\lambda_s}(\hat{\beta}_j)\times\frac{\partial\hat{\beta}_j}{\partial\lambda_t} + \frac{\partial \dddot{r}_{j,\boldsymbol{\lambda}}}{\partial\lambda_t}(\hat{\beta}_j)\times\frac{\partial\hat{\beta}_j}{\partial\lambda_s} \\
&\quad + \dddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j)\times\frac{\partial^2\hat{\beta}_j}{\partial\lambda_s\partial\lambda_t} + \ddddot{r}_{j,\boldsymbol{\lambda}}(\hat{\beta}_j)\times\frac{\partial\hat{\beta}_j}{\partial\lambda_s}\times\frac{\partial\hat{\beta}_j}{\partial\lambda_t}.
\end{aligned}$$

We omit the steps for the other derivatives as they are straightforward.

## References


D. M. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16:125–127, 1974.

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

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

C. B. Do, D. A. Woods, A. Y. Ng. Efficient multiple hyperparameter learning for log-linear models. *Advances in Neural Information Processing Systems*, 20:377–384, 2008.

W. J. Fu. Penalized regression: the bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7:397–416, 1998.

G. C. McDonald and R. C. Schwing. Instabilities of regression estimates relating air pollution to mortality. *Technometrics*, 15:463–481, 1973.

J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

J. Nocedal and S. J. Wright. Numerical Optimization. *Springer Series in Operations Research and Financial Engineering*. Springer, Second edition, 1999.

K. R. Rad and A. Maleki. A scalable estimate of the extra-sample prediction error via approximate leave-one-out. *ArXiv:1801.10243v4*, 2020.

K. R. Rad, W. Zhou, A. Maleki. Error bounds in estimating the out-of-sample prediction error using leave-one-out cross validation in high-dimensions. *ArXiv:2003.01770v1*, 2020.

D. C. Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, 1982.