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 (LO) can be efficiently estimated with an approximate leave-one-out formula (ALO). We consider the problem of adjusting hyperparameters so as to maximize performance on 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 demonstate the usefulness of the approach by finding hyperparameters for regularized logistic regression on a variety of different real-world problems.

Keywords: Hyperparameter Optimization, Logistic Regression, Ridge Regression, Machine Learning, Python

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 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 ℓ is a loss function, $R_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) = \sum_{j=1}^{p} r_{\boldsymbol{\lambda}}(\beta_j)$ is a regularizer, and $\boldsymbol{\lambda}$ are 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 mimize the out-of-sample prediction error

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

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 $\mathrm{Err}_{\mathrm{out}}$ (Rad et al., 2020). In the general case, LO can be expensive to compute, requiring a model to be fit n times; but under certain conditions, it can be efficiently estimated with 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). Additionally, 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 optimizer to find hyperparameters for several different classes of regularized models.

1.1 Relevant Work

Grid search is a commonly used approach for hyperparameter optimization (HO). While 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 (Do et al., 2008; Bengio, 2000). By using ALO, we expect our objective to be a more accurate proxy for $\mathrm{Err}_{\mathrm{out}}$. We further improve on previous approaches by computing the objective hessian with respect to hyperparameters and applying trust-region optimization to ensure we find a local minimum.

1.2 Notation

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

2. Preliminary: The ALO Formula

The LO estimate for $\mathrm{Err}_{\mathrm{out}}$ is defined as

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

where

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

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 (2) at $\hat{\boldsymbol{\beta}}$. Let $\boldsymbol{H}$ denote the hessian of (1) at $\hat{\boldsymbol{\beta}}$; and let $\boldsymbol{g}_{/i}$ and $\boldsymbol{H}_{/i}$ denote the gradient and hessian of (2) at $\hat{\boldsymbol{\beta}}$. Define

$$\begin{aligned} \ell_i\left(u\right) &\triangleq \ell\left(y_i \mid u\right) \\ u_i &\triangleq \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}} \\ \boldsymbol{W} &\triangleq \nabla^2 R_{\boldsymbol{\lambda}}(\hat{\boldsymbol{\beta}}) \end{aligned}$$

Then

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

where $\boldsymbol{A}$ is a diagonal matrix with $\boldsymbol{A}_{ii} = \ddot{\ell}_i(u_i)$ and $\boldsymbol{X}$ represents the feature matrix whose i$^{\text{th}}$ row is $\boldsymbol{x}_i$. For $\boldsymbol{g}_{/i}$ and $\boldsymbol{H}_{/i}$, we have

$$\begin{aligned}\boldsymbol{g}_{/i} &= -\dot{\ell}_i(u_i)\,\boldsymbol{x}_i \\ \boldsymbol{H}_{/i} &= \boldsymbol{H} - \ddot{\ell}_i(u_i)\,\boldsymbol{x}_i\boldsymbol{x}_i^\top\end{aligned}$$

The ALO approximation to $\hat{\boldsymbol{\beta}}_{/i}$ is then

$$\tilde{\boldsymbol{\beta}}_{/i} \triangleq \hat{\boldsymbol{\beta}} - \boldsymbol{H}_{/i}^{-1}\boldsymbol{g}_{/i}$$

Computed naively, this formula would require inverting a p-by-p matrix n times, but we can apply the matrix inversion lemma to derive a more efficient formula using $\boldsymbol{H}^{-1}$. Put $h_i \triangleq \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\boldsymbol{x}_i$. Then

$$\begin{aligned}\boldsymbol{x}_i^\top \tilde{\boldsymbol{\beta}}_{/i} &= \boldsymbol{x}_i^\top \left[\hat{\boldsymbol{\beta}} - \boldsymbol{H}_{/i}^{-1}\boldsymbol{g}_{/i}\right] \\ &= u_i - \boldsymbol{x}_i^\top \left[\boldsymbol{H} - \ddot{\ell}_i(u_i)\,\boldsymbol{x}_i\boldsymbol{x}_i^\top\right]^{-1}\left(-\dot{\ell}_i(u_i)\,\boldsymbol{x}_i\right) \\ &= u_i + \dot{\ell}_i(u_i)\,\boldsymbol{x}_i^\top \left[\boldsymbol{H} - \ddot{\ell}_i(u_i)\,\boldsymbol{x}_i\boldsymbol{x}_i^\top\right]^{-1}\boldsymbol{x}_i \\ &= u_i + \dot{\ell}_i(u_i)\,\boldsymbol{x}_i^\top \left[\boldsymbol{H}^{-1} + \frac{\ddot{\ell}_i(u_i)\,\boldsymbol{H}^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^\top\boldsymbol{H}^{-1}}{1 - \ddot{\ell}_i(u_i)\,\boldsymbol{x}_i^\top\boldsymbol{H}^{-1}\boldsymbol{x}_i}\right]\boldsymbol{x}_i \\ &= u_i + \dot{\ell}_i(u_i)\,h_i\left[1 + \frac{\ddot{\ell}_i(u_i)\,h_i}{1 - \ddot{\ell}_i(u_i)\,h_i}\right] \\ &= u_i + \frac{\dot{\ell}_i(u_i)\,h_i}{1 - \ddot{\ell}_i(u_i)\,h_i}\end{aligned}$$

And ALO can be defined as

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

3. Optimizing ALO

Put

$$\begin{aligned}\tilde{u}_i(u,h) &\triangleq u + \frac{\dot{\ell}_i(u)\,h}{1 - \ddot{\ell}_i(u)\,h} \\ \tilde{\ell}_i(u,h) &\triangleq \ell_i\left(\tilde{u}_i(u,h)\right) \\ f_i(\boldsymbol{\lambda}) &\triangleq \tilde{\ell}_i(u_i,h_i) \\ f(\boldsymbol{\lambda}) &\triangleq \sum_{i=1}^{n} f_i(\boldsymbol{\lambda})\end{aligned}$$

Minimizing f is equivalent to minimizing $\text{ALO}_{\boldsymbol{\lambda}}$. By deriving equations for ∇f and $\nabla^2 f$, we can apply a second-order optimizer to find a local minimum. Though there's no guarantee that $\nabla^2 f$ is positive-definite, such optimization problems can be handled well by trust-region optimizers (Sorensen, 1982).

3.1 ALO Gradient

Differentiating f gives us

$$\frac{\partial f}{\partial \lambda_s} = \sum_{i=1}^{n} \frac{\partial f_i}{\partial \lambda_s}$$

and

$$\frac{\partial f_i}{\partial \lambda_s} = \frac{\partial \tilde{\ell}_i}{\partial u}(u_i, h_i) \times \frac{\partial u_i}{\partial \lambda_s} + \frac{\partial \tilde{\ell}_i}{\partial h}(u_i, h_i) \times \frac{\partial h_i}{\partial \lambda_s}$$

For the derivative of u_i, we get

$$\frac{\partial u_i}{\partial \lambda_s} = \boldsymbol{x}_i^\top \frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_s}$$

To derive the derivative of $\hat{\boldsymbol{\beta}}$, we observe that at the optimum of (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

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

For the derivative of h_i, we have

$$\frac{\partial h_i}{\partial \lambda_s} = \frac{\partial}{\partial \lambda_s}\left(\boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \boldsymbol{x}\right) = -\boldsymbol{x}_i^\top \boldsymbol{H}^{-1} \frac{\partial \boldsymbol{H}}{\partial \lambda_s} \boldsymbol{H}^{-1} \boldsymbol{x}_i$$

and

$$\begin{aligned}
\frac{\partial \boldsymbol{H}}{\partial \lambda_s} &= \frac{\partial}{\partial \lambda_s}\left(\boldsymbol{X}^\top \boldsymbol{A} \boldsymbol{X} + \boldsymbol{W}\right) = \boldsymbol{X}^\top \frac{\partial \boldsymbol{A}}{\partial \lambda_s} \boldsymbol{X} + \frac{\partial \boldsymbol{W}}{\partial \lambda_s} \\
\left(\frac{\partial \boldsymbol{A}}{\partial \lambda_s}\right)_{ii} &= \dddot{\ell}_i\left(u_i\right) \times \frac{\partial u_i}{\partial \lambda_s} \\
\left(\frac{\partial \boldsymbol{W}}{\partial \lambda_s}\right)_{jj} &= \frac{\partial}{\partial \lambda_s}\left(\ddot{r}_{\boldsymbol{\lambda}}(\hat{\beta}_j)\right) = \frac{\partial \ddot{r}_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\beta}_j) + \dddot{r}_{\boldsymbol{\lambda}}(\hat{\beta}_j) \times \frac{\partial \hat{\beta}_j}{\partial \lambda_s}
\end{aligned}$$

Differentiating $\tilde{\ell}_i$ gives us

$$\frac{\partial \tilde{\ell}_i}{\partial u} = \frac{\partial}{\partial u}\left(\ell_i\left(\tilde{u}_i\left(u,h\right)\right)\right) = \dot{\ell}_i\left(\tilde{u}_i\left(u,h\right)\right) \times \frac{\partial \tilde{u}_i}{\partial u}$$
$$\frac{\partial \tilde{\ell}_i}{\partial h} = \frac{\partial}{\partial h}\left(\ell_i\left(\tilde{u}_i\left(u,h\right)\right)\right) = \dot{\ell}_i\left(\tilde{u}_i\left(u,h\right)\right) \times \frac{\partial \tilde{u}_i}{\partial h}$$

and

$$\frac{\partial \tilde{u}_i}{\partial u} = \frac{\partial}{\partial u}\left(u + \frac{\dot{\ell}_i\left(u\right)h}{1-\ddot{\ell}_i\left(u\right)h}\right) = 1 + \frac{\ddot{\ell}_i\left(u\right)h}{1-\ddot{\ell}_i\left(u\right)h} + \frac{\dot{\ell}_i\left(u\right)\dddot{\ell}_i\left(u\right)h^2}{\left(1-\ddot{\ell}_i\left(u\right)h\right)^2}$$
$$\frac{\partial \tilde{u}_i}{\partial h} = \frac{\partial}{\partial h}\left(u + \frac{\dot{\ell}_i\left(u\right)h}{1-\ddot{\ell}_i\left(u\right)h}\right) = \frac{\dot{\ell}_i\left(u\right)}{\left(1-\ddot{\ell}_i\left(u\right)h\right)^2}$$

3.2 ALO Hessian

The ALO hessian terms break down as

$$\frac{\partial^2 f}{\partial \lambda_s \partial \lambda_t} = \sum_{i=1}^{n} \frac{\partial^2 f_i}{\partial \lambda_s \partial \lambda_t}$$

and

$$\begin{aligned}
\frac{\partial^2 f_i}{\partial \lambda_s \partial \lambda_t} &= \frac{\partial}{\partial \lambda_t}\left[\frac{\partial \tilde{\ell}_i}{\partial u}(u_i,h_i) \times \frac{\partial u_i}{\partial \lambda_s} + \frac{\partial \tilde{\ell}_i}{\partial h}(u_i,h_i) \times \frac{\partial h_i}{\partial \lambda_s}\right] \\
&= \frac{\partial \tilde{\ell}_i}{\partial u}(u_i,h_i) \times \frac{\partial^2 u_i}{\partial \lambda_s \partial \lambda_t} + \frac{\partial^2 \tilde{\ell}_i}{\partial u^2}(u_i,h_i) \times \frac{\partial u_i}{\partial \lambda_s} \times \frac{\partial u_i}{\partial \lambda_t} \\
&+ \frac{\partial^2 \tilde{\ell}_i}{\partial u \partial h}(u_i,h_i) \times \left[\frac{\partial u_i}{\partial \lambda_s} \times \frac{\partial h_i}{\partial \lambda_t} + \frac{\partial u_i}{\partial \lambda_t} \times \frac{\partial h_i}{\partial \lambda_s}\right] \\
&+ \frac{\partial \tilde{\ell}_i}{\partial h}(u_i,h_i) \times \frac{\partial^2 h_i}{\partial \lambda_s \partial \lambda_t} + \frac{\partial^2 \tilde{\ell}_i}{\partial h^2}(u_i,h_i) \times \frac{\partial h_i}{\partial \lambda_s} \times \frac{\partial h_i}{\partial \lambda_t}
\end{aligned}$$

Starting with the second derivative of u_i, we have

$$\frac{\partial^2 u_i}{\partial \lambda_s \partial \lambda_t} = \boldsymbol{x}_i^\top \frac{\partial^2 \hat{\boldsymbol{\beta}}}{\partial \lambda_s \partial \lambda_t}$$

and

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

Now,

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

and

$$\frac{\partial}{\partial \lambda_t}\left(\frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\right) = \frac{\partial \nabla^2 R_{\boldsymbol{\lambda}}}{\partial \lambda_s}(\hat{\boldsymbol{\beta}})\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \lambda_t} + \frac{\partial \nabla R_{\boldsymbol{\lambda}}}{\partial \lambda_s \partial \lambda_t}(\hat{\boldsymbol{\beta}})$$

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

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

For the second derivative of h_i, we derive

$$\begin{aligned}\frac{\partial^2 h_i}{\partial \lambda_s \partial \lambda_t} &= \frac{\partial}{\partial \lambda_t}\left(-\boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{H}^{-1}\boldsymbol{x}_i\right) \\ &= \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_t}\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{H}^{-1}\boldsymbol{x}_i + \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i \\ &- \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial^2 \boldsymbol{H}}{\partial \lambda_s \partial \lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i \\ &= 2\boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{H}^{-1}\frac{\partial^2 \boldsymbol{H}}{\partial \lambda_s \partial \lambda_t}\boldsymbol{H}^{-1}\boldsymbol{x}_i\end{aligned}$$

and

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

Finally, for the second derivatives of $\tilde{\ell}_i$, we have

$$
\begin{aligned}
\frac{\partial^2 \tilde{\ell}_i}{\partial u^2} &= \frac{\partial}{\partial u}\left(\dot{\ell}_i(\tilde{u}_i(u,h)) \times \frac{\partial \tilde{u}_i}{\partial u}\right) \\
&= \dot{\ell}_i(\tilde{u}_i(u,h)) \times \frac{\partial^2 \tilde{u}_i}{\partial u^2} + \ddot{\ell}_i(\tilde{u}_i(u,h)) \times \left(\frac{\partial \tilde{u}_i}{\partial u}\right)^2 \\
\frac{\partial^2 \tilde{\ell}_i}{\partial u \partial h} &= \frac{\partial}{\partial u}\left(\dot{\ell}_i(\tilde{u}_i(u,h)) \times \frac{\partial \tilde{u}_i}{\partial h}\right) \\
&= \dot{\ell}_i(\tilde{u}_i(u,h)) \times \frac{\partial^2 \tilde{u}_i}{\partial u \partial h} + \ddot{\ell}_i(\tilde{u}_i(u,h)) \times \frac{\partial \tilde{u}_i}{\partial u} \times \frac{\partial \tilde{u}_i}{\partial h} \\
\frac{\partial^2 \tilde{\ell}_i}{\partial h^2} &= \frac{\partial}{\partial h}\left(\dot{\ell}_i(\tilde{u}_i(u,h)) \times \frac{\partial \tilde{u}_i}{\partial h}\right) \\
&= \dot{\ell}_i(\tilde{u}_i(u,h)) \times \frac{\partial^2 \tilde{u}_i}{\partial h^2} + \ddot{\ell}_i(\tilde{u}_i(u,h)) \times \left(\frac{\partial \tilde{u}_i}{\partial h}\right)^2
\end{aligned}
$$

and

$$\begin{aligned}
\frac{\partial^2 \tilde{u}_i}{\partial u^2} &= \frac{\partial}{\partial u}\left(1 + \frac{\ddot{\ell}_i(u)\,h}{1-\ddot{\ell}_i(u)\,h} + \frac{\dot{\ell}_i(u)\,\dddot{\ell}_i(u)\,h^2}{(1-\ddot{\ell}_i(u)\,h)^2}\right) \\
&= \frac{\dddot{\ell}_i(u)\,h}{(1-\ddot{\ell}_i(u)\,h)^2} + \frac{(\ddot{\ell}_i(u)\,\dddot{\ell}_i(u) + \dot{\ell}_i(u)\,\ddddot{\ell}_i(u))h^2}{(1-\ddot{\ell}_i(u)\,h)^2} + \frac{2\dot{\ell}_i(u)\,\dddot{\ell}_i(u)^2\,h^3}{(1-\ddot{\ell}_i(u)\,h)^3} \\
\frac{\partial^2 \tilde{u}_i}{\partial u \partial h} &= \frac{\partial}{\partial u}\left(\frac{\dot{\ell}_i(u)}{(1-\ddot{\ell}_i(u)\,h)^2}\right) = \frac{\ddot{\ell}_i(u)}{(1-\ddot{\ell}_i(u)\,h)^2} + \frac{2\dot{\ell}_i(u)\,\dddot{\ell}_i(u)\,h}{(1-\ddot{\ell}_i(u)\,h)^3} \\
\frac{\partial^2 \tilde{u}_i}{\partial h^2} &= \frac{\partial}{\partial h}\left(\frac{\dot{\ell}_i(u)}{(1-\ddot{\ell}_i(u)\,h)^2}\right) = \frac{2\dot{\ell}_i(u)\,\ddot{\ell}_i(u)}{(1-\ddot{\ell}_i(u)\,h)^3}
\end{aligned}$$

3.3 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

$$\begin{aligned}
\boldsymbol{t}_i &= {\boldsymbol{L}^{-1}}^\top \boldsymbol{L}^{-1}\boldsymbol{x}_i \\
\frac{\partial h_i}{\partial \lambda_s} &= \boldsymbol{t}_i^\top \frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{t}_i
\end{aligned}$$

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

$$\begin{aligned}
\boldsymbol{r}_{si} &= \boldsymbol{L}^{-1}\frac{\partial \boldsymbol{H}}{\partial \lambda_s}\boldsymbol{H}^{-1}\boldsymbol{x}_i \\
\frac{\partial^2 h_i}{\partial \lambda_s \partial \lambda_t} &= \boldsymbol{r}_{si}^\top \boldsymbol{r}_{ti} - \boldsymbol{t}_i^\top \frac{\partial^2 \boldsymbol{H}}{\partial \lambda_s \partial \lambda_t}\boldsymbol{t}_i
\end{aligned}$$

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 $\mathcal{O}\left(n^2 qp\right)$ 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.

4. Numerical Experiments

We apply ALO optimization to regularized logistic regression on a range of public real-world binary classification data sets (Table 1) and explore its utility for tunning one or more hyperparameters. For the data set `Breast Cancer`, the objective is to predict whether

Data Set	**n**	**p**
`Breast Cancer`	569	30
`Cleveland Heart`	297	22
`USPS 35`	1540	256
`Arcene`	200	10000
`Gisette`	7000	5000

Table 1

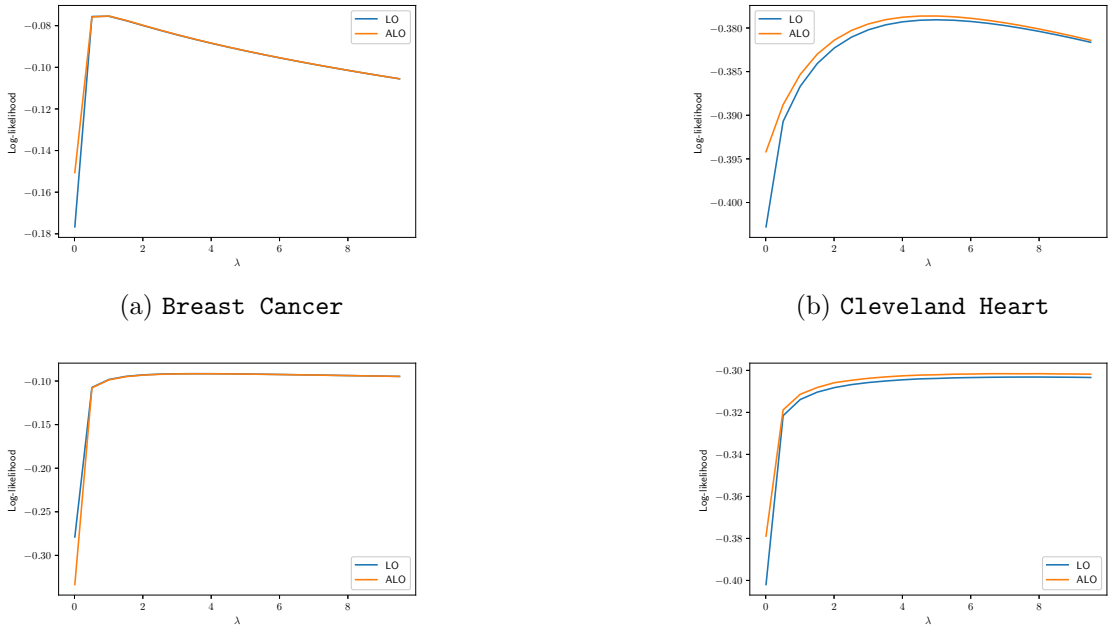
breast mass is malignant from characteristics of cell nuclei. For `Cleveland Heart`, the objective is to detect the presence of heart disease. The data set uses 13 features, but 5 are categorical. We obtain 22 features after transforming the categorical features to indicator variables, and we obtain 297 entries after dropping any entries with missing features. For `USPS 35` the objective is to predict whether a 16x16 pixel image represents the digit 3 or the digit 5. It is a subset of the original 10 digit USPS classification problem. For `Arcene`, 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. And for `Gisette`, the objective is to predict whether a handwritten digit is a 4 or a 9. The dataset 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.

4.1 One Hyperparameter

We begin by fitting models with a single hyperparameter regularizer $R_\lambda\left(\boldsymbol{\beta}\right) = \lambda\|\boldsymbol{\beta}\|^2$. We compare the ALO approach to a standard grid search. For ALO, we use the Python package `peak-engines` available from `https://github.com/rnburn/peak-engines` and

	ALO Optimization				Grid Search			
Data Set	**Time**	λ	**ALO**	**LO**	**Time**	λ	**ALO**	**LO**
`Breast Cancer`	0.014s	0.751	-0.0749	-0.0749	0.46s	1.39	-0.0769	-0.0770
`Cleveland Heart`	0.0033s	4.79	-0.379	-0.379	0.11s	1.39	-0.383	-0.385
`USPS 35`	0.34s	3.69	-0.0917	-0.0916	0.87s	10.8	-0.0955	-0.0955
`Arcene`	1.6s	7.23	-0.302	-0.303	4.2s	10.8	-0.302	-0.304

Table 2



(a) `Breast Cancer` (b) `Cleveland Heart`

(c) `USPS 35` (d) `Arcene`

Figure 1: ALO and LO log-likelihoods

for grid search we use `LogisticRegressionCV` from `sklearn-0.23.1` with default parameters, which runs a 5-fold CV for 10 different hyperparameters across a logarithmically spaced grid. Table 2 summarizes the results of the experiment: it shows how long the hyperparameter search took in seconds, the best hyperparameter found λ, the ALO log-likelihood for λ, and the LO log-likelihood for λ, computed by brute force. To visualize how closely ALO tracks LO, we compute the values across a range of λ and plot the results (Figure 1).

4.2 Multiple Hyperparameters

To test ALO optimization with multiple hyperparameters, we fit bridge regularizers (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 $r_{\boldsymbol{\lambda}}$ has a continuous fourth derivative, we interpolate the regularizers

	Bridge Regularization		Ridge Regularization	
Fold	$\boldsymbol{\lambda}$	**Mean Log-likelihood**	$\boldsymbol{\lambda}$	**Mean Log-likelihood**
0	[17.3, 1.95]	-0.0698	19.1	-0.0709
1	[16.8, 1.94]	-0.0581	18.7	-0.0587
2	[13.0, 1.81]	-0.0667	18.3	-0.0697
3	[10.5, 1.75]	-0.0677	16.5	-0.0714
4	[10.8, 1.83]	-0.0639	14.4	-0.0667
Mean		-0.0652		-0.0675

Table 3: 5-Fold Cross-validation on `Gisette` data set

with a polynomial when $|\beta_j| < \delta$ so that

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

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 δ^{λ_2} up to the fourth derivative. For our experiment, we use $\delta = 0.01$. For comparison, we also fit models with ridge regularization. Table 3 shows the hyperparameters found for bridge regularization and ridge regularization and compares their performance on each cross-validation fold.

5. Discussion

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 logistic regression models to a variety of different real-world data sets. We found that even when using a single-parameter regularizer, we were able to find hyperparameters with better LO scores 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.

Acknowledgments

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

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: 1889–1900, 2000, 12:1889–1900, 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.

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.