On Optimal Generalizability in Parametric Learning

Ahmad Beirami∗
beirami@seas.harvard.edu

Meisam Razaviyayn†
razaviya@usc.edu

Shahin Shahrampour∗
shahin@seas.harvard.edu

Vahid Tarokh∗
vahid@seas.harvard.edu

Abstract

We consider the parametric learning problem, where the objective of the learner is determined by a parametric loss function. Employing empirical risk minimization with possibly regularization, the inferred parameter vector will be biased toward the training samples. Such bias is measured by the cross validation procedure in practice where the data set is partitioned into a training set used for training and a validation set, which is not used in training and is left to measure the out-of-sample performance. A classical cross validation strategy is the leave-one-out cross validation (LOOCV) where one sample is left out for validation and training is done on the rest of the samples that are presented to the learner, and this process is repeated on all of the samples. LOOCV is rarely used in practice due to the high computational complexity. In this paper, we first develop a computationally efficient approximate LOOCV (ALOOCV) and provide theoretical guarantees for its performance. Then we use ALOOCV to provide an optimization algorithm for finding the regularizer in the empirical risk minimization framework. In our numerical experiments, we illustrate the accuracy and efficiency of ALOOCV as well as our proposed framework for the optimization of the regularizer.

1 Introduction

We consider the parametric supervised/unsupervised learning problem, where the objective of the learner is to build a predictor based on a set of historical data. Let $z^n = \{z_i\}_{i=1}^n$, where $z_i \in \mathcal{Z}$ denotes the data samples at the learner’s disposal that are assumed to be drawn i.i.d. from an unknown density function $p(\cdot)$, and $\mathcal{Z}$ is compact.

We assume that the learner expresses the objective in terms of minimizing a parametric loss function $\ell(z; \theta)$, which is a function of the parameter vector $\theta$. The learner solves for the unknown parameter vector $\theta^* \in \Theta \subseteq \mathbb{R}^k$, where $k$ denotes the number of parameters in the model class, and $\Theta$ is a convex, compact set.

Let

$$\mathcal{L}(\theta) \triangleq E\{\ell(z; \theta)\}$$

be the risk associated with the parameter vector $\theta$, where the expectation is with respect to the density $p(\cdot)$ that is unknown to the learner. Ideally, the goal of the learner is to choose the parameter vector $\theta^*$ such that $\theta^* = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta) = \arg\min_{\theta \in \Theta} E\{\ell(z; \theta)\}$. Since the density function $p(\cdot)$ is unknown, the learner cannot compute $\theta^*$ and hence cannot achieve the ideal performance of $\mathcal{L}(\theta^*) = \min_{\theta \in \Theta} \mathcal{L}(\theta)$ associated with the model class $\Theta$. Instead, one can consider the minimiza-

∗School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA.
†Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089, USA.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
A main application of cross validation (see [1] for a recent survey) is in model selection. Related work: bootstrap techniques.

Most of the existing methods, on the other hand, ignore the response and carry out the optimization of the single parameter in ridge regression where cross validation can be obtained in closed form [16]. Our second contribution is a gradient descent optimization algorithm for tuning the regularization hyperparameters in parametric learning problems. A similar approach has been taken for tuning hyperparameters of predicting overfitting and hence can be used for model selection and choosing the regularization hyperparameter. Finally, we expect that the idea of ALOOCV can be extended to derive computationally efficient approximate bootstrap estimators [15].

Contribution: Our first contribution is to provide an approximation for the cross validation vector, called ALOOCV, with much lower computational cost. We compare its performance with LOOCV in problems of reasonable size where LOOCV is tractable. We also test it on problems of large size where LOOCV is practically impossible to implement. We describe how to handle quasi-smooth loss/regularizer functions. We also show that ALOOCV is asymptotically equivalent to Takeuchi information criterion (TIC) under certain regularity conditions.

Our second contribution is to use ALOOCV to develop a gradient descent algorithm for jointly optimizing the regularization hyperparameters as well as the unknown parameter vector $\theta$. We show that multiple hyperparameters could be tuned using the developed algorithm. We emphasize that the second contribution would not have been possible without the developed estimator as obtaining the gradient of the LOOCV with respect to tuning parameters is computationally expensive. Our experiments show that the developed method handles quasi-smooth regularized loss functions as well as number of tuning parameters that is on the order of the training samples.

Finally, it is worth mentioning that although the leave-one-out cross validation scenario is considered in our analyses, the results and the algorithms can be extended to the leave-$q$-out cross validation and bootstrap techniques.

Related work: A main application of cross validation (see [1] for a recent survey) is in model selection [2,3]. On the theoretical side, the proposed approximation on LOOCV is asymptotically equivalent to Takeuchi information criterion (TIC) [4,7], under certain regularity conditions (see [8] for a proof of asymptotic equivalence of AIC and LOOCV in autoregressive models). This is also related to Barron’s predicted square error (PSE) [9] and Moody’s effective number of parameters for nonlinear systems [10]. Despite these asymptotic equivalences our main focus is on the non-asymptotic performance of ALOOCV.

ALOOCV simplifies to the closed form derivation of the LOOCV for linear regression, called PRESS (see [11,12]). Hence, this work can be viewed as an approximate extension of this closed form derivation for an arbitrary smooth regularized loss function. This work is also related to the concept of influence functions [13], which has recently received renewed interest [14]. In contrast to methods based on influence functions that require large number of samples due to their asymptotic nature, we empirically show that the developed ALOOCV works well even when the number of samples and features are small and comparable to each other. In particular, ALOOCV is capable of predicting overfitting and hence can be used for model selection and choosing the regularization hyperparameter. Finally, we expect that the idea of ALOOCV can be extended to derive computationally efficient approximate bootstrap estimators [15].

Our second contribution is a gradient descent optimization algorithm for tuning the regularization hyperparameters in parametric learning problems. A similar approach has been taken for tuning the single parameter in ridge regression where cross validation can be obtained in closed form [16]. Most of the existing methods, on the other hand, ignore the response and carry out the optimization solely based on the features, e.g., Stein unbiased estimator of the risk for multiple parameter selection [17,18].
Bayesian optimization has been used for tuning the hyperparameters in the model [19–23], which postulates a prior on the parameters and optimizes for the best parameter. Bayesian optimization methods are generally derivative free leading to slow convergence rate. In contrast, the proposed method is based on a gradient descent method. Other popular approaches to the tuning of the optimization parameters include grid search and random search [24–26]. These methods, by nature, also suffer from slow convergence. Finally, model selection has been considered as a bi-level optimization [27–28] where the training process is modeled as a second level optimization problem within the original problem. These formulations, similar to many other bi-level optimization problems, often lead to computationally intensive algorithms that are not scalable.

We remark that ALOOCV can also be used within Bayesian optimization, random search, and grid search methods. Further, resource allocation can be used for improving the optimization performance in all of such methods.

2 Problem Setup

To facilitate the presentation of the ideas, let us define the following concepts. Throughout, we assume that all the vectors are in column format.

**Definition 1 (regularization vector/regularized loss function)** We suppose that the learner is concerned with M regularization functions \( r_1(\theta), \ldots, r_M(\theta) \) in addition to the main loss function \( \ell(z; \theta) \). We define the regularization vector \( r(\theta) \) as
\[
 r(\theta) \triangleq (r_1(\theta), \ldots, r_M(\theta))^\top.
\]

Further, let \( \lambda = (\lambda_1, \ldots, \lambda_M)^\top \) be the vector of regularization parameters. We call \( w_n(z; \theta, \lambda) \) the regularized loss function given by
\[
 w_n(z; \theta, \lambda) \triangleq \ell(z; \theta) + \frac{1}{n} \lambda^\top r(\theta) = \ell(z; \theta) + \frac{1}{n} \sum_{m=1}^{M} \lambda_m r_m(\theta).
\]

The above definition encompasses many popular learning problems. For example, elastic net regression [31] can be cast in this framework by setting \( r_1(\theta) = \|\theta\|_1 \) and \( r_2(\theta) = \frac{1}{2} \|\theta\|^2_2 \).

**Definition 2 (empirical risk/regularized empirical risk)** Let the empirical risk be defined as \( \hat{L}_{\cdot,n}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(z_i; \theta) \). Similarly, let the regularized empirical risk be defined as \( \hat{\mathcal{W}}_{\cdot,n}(\theta, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \{w_n(z_i; \theta, \lambda)\} \).

**Definition 3 (regularized empirical risk minimization)** We suppose that the learner solves the empirical risk minimization problem by selecting \( \hat{\theta}_\lambda(z^n) \) as follows:
\[
 \hat{\theta}_\lambda(z^n) \in \arg \min_{\theta \in \Theta} \left\{ \hat{\mathcal{W}}_{\cdot,n}(\theta, \lambda) \right\} = \arg \min_{\theta \in \Theta} \left\{ \sum_{i \in [n]} \ell(z_i; \theta) + \lambda^\top r(\theta) \right\}.
\]

Once the learner solves for \( \hat{\theta}_\lambda(z^n) \), the empirical risk corresponding to \( \hat{\theta}_\lambda(z^n) \) can be readily computed by \( \hat{L}_{\cdot,n}(\hat{\theta}_\lambda(z^n)) = \frac{1}{n} \sum_{i \in [n]} \ell(z_i; \hat{\theta}_\lambda(z^n)) \). While the learner can evaluate her performance on the observed data samples (also called the in-sample empirical risk, i.e., \( \hat{L}_{\cdot,n}(\hat{\theta}_\lambda(z^n)) \)), it is imperative to assess the performance of the learner on unobserved fresh samples, i.e., \( \mathcal{L}(\hat{\theta}_\lambda(z^n)) \) (see (1)), which is referred to as the out-of-sample risk. To measure the out-of-sample risk, it is a common practice to perform cross validation as it works outstandingly well in many practical situations and is conceptually universal and simple to implement.

Leave-one-out cross validation (LOOCV) uses all of the samples but one for training, which is left out for testing, leading to an n-dimensional cross validation vector of out-of-sample estimates. Let us formalize this notion. Let \( z^{(i)} \triangleq (z_{1}, \ldots, z_{i-1}, z_{i+1}, \ldots, z_{n}) \) denote the set of the training examples excluding \( z_i \).
Definition 4 (LOOCV empirical risk minimization/cross validation vector) Let $\hat{\theta}_\lambda(z^{n\setminus i})$ be the estimated parameter over the training set $z^{n\setminus i}$, i.e.,

$$\hat{\theta}_\lambda(z^{n\setminus i}) \in \arg \min_{\theta \in \mathbb{R}^p} \left\{ \hat{W}_{z^{n\setminus i}}(\theta, \lambda) \right\} = \arg \min_{\theta \in \mathbb{R}^p} \left\{ \sum_{j \in [n] \setminus i} \ell(z_j; \theta) + \lambda^\top r(\theta) \right\}. \quad (3)$$

The cross validation vector is given by $\{CV_{\lambda,i}(z^n)\}_{i \in [n]}$ where $CV_{\lambda,i}(z^n) \triangleq \ell(z_i; \hat{\theta}_\lambda(z^{n\setminus i}))$, and the cross validation out-of-sample estimate is given by $CV_{\lambda}(z^n) \triangleq \frac{1}{n} \sum_{i=1}^n CV_{\lambda,i}(z^n)$.

The empirical mean and the empirical variance of the $n$-dimensional cross validation vector are used by practitioners as surrogates on assessing the out-of-sample performance of a learning method.

The computational cost of solving the problem in (3) is $n$ times that of the original problem in (2). Hence, while LOOCV provides a simple yet powerful tool to estimate the out-of-sample performance, the additional factor of $n$ in terms of the computational cost makes LOOCV impractical in large-scale problems. One common solution to this problem is to perform validation on fewer number of samples, where the downside is that the learner would obtain a much more noisy and sometimes completely unstable estimate of the out-of-sample performance compared to the case where the entire LOOCV vector is at the learner’s disposal. On the other hand, ALOOCV described next will provide the benefits of LOOCV with negligible additional computational cost.

We emphasize that the presented problem formulation is general and includes a variety of parametric machine learning tasks, where the learner empirically solves an optimization problem to minimize some loss function.

### 3 Approximate Leave-One-Out Cross Validation (ALOOCV)

We assume that the regularized loss function is three times differentiable with continuous derivatives (see Assumption 1). This includes many learning problems, such as the $L_2$ regularized logistic loss function. We later comment on how to handle the $\ell_1$ regularizer function in LASSO. To proceed, we need one more definition.

**Definition 5 (Hessian/empirical Hessian)** Let $H(\theta)$ denote the Hessian of the risk function defined as $H(\theta) \triangleq \nabla^2 \mathcal{L}(\theta)$. Further, let $\hat{H}_{z^n}(\theta, \lambda)$ denote the empirical Hessian of the regularized loss function, defined as $\hat{H}_{z^n}(\theta, \lambda) \triangleq \hat{E}_{z^n} \left\{ \nabla^2 w_n(z; \theta, \lambda) \right\} = \frac{1}{n} \sum_{i=1}^n \nabla^2 w_n(z_i; \theta, \lambda)$. Similarly, we define $\hat{H}_z(\theta, \lambda) \triangleq \hat{E}_{z^{n\setminus i}} \left\{ \nabla^2 w_n(z; \theta, \lambda) \right\} = \frac{1}{n-1} \sum_{i \in [n] \setminus i} \nabla^2 w_n(z_i; \theta, \lambda)$.

Next we present the set of assumptions we need to prove the main result of the paper.

**Assumption 1** We assume that

(a) There exists $\theta^* \in \Theta$ such that $\|\hat{\theta}_\lambda(z^n) - \theta^*\|_\infty = o_p(1)$

(b) $w_n(z; \theta)$ is of class $C^3$ as a function of $\theta$ for all $z \in \mathcal{Z}$.

(c) $H(\theta^*) \succ 0$ is positive definite.

**Theorem 1** Under Assumption 1 let

$$\hat{\theta}_{\lambda}^{(i)}(z^n) \triangleq \hat{\theta}_\lambda(z^n) + \frac{1}{n-1} \left( \hat{H}_{z^n\setminus i} \left( \hat{\theta}_\lambda(z^n), \lambda \right) \right)^{-1} \nabla_\theta \ell(z_i; \hat{\theta}_\lambda(z^n)), \quad (4)$$

assuming the inverse exists. Then,

$$\hat{\theta}_\lambda(z^{n\setminus i}) - \hat{\theta}_{\lambda}^{(i)}(z^n) = \frac{1}{n-1} \left( \hat{H}_{z^n\setminus i} \left( \hat{\theta}_\lambda(z^n), \lambda \right) \right)^{-1} \varepsilon_{\lambda,n}^{(i)}, \quad (5)$$

---

1. $\Theta^\circ$ denotes the interior operator.
2. $X_n = o_p(a_n)$ implies that $X_n/a_n$ approaches 0 in probability with respect to the density function $p(\cdot)$. 

---

4
with high probability where
\[ \varepsilon^{(i)}_{\lambda,n} = \varepsilon^{(i),1}_{\lambda,n} - \varepsilon^{(i),2}_{\lambda,n}, \]  
and \( \varepsilon^{(i),1}_{\lambda,n} \) is defined as
\[ \varepsilon^{(i),1}_{\lambda,n} \triangleq \frac{1}{2} \sum_{j \in [n] \setminus i} \sum_{k \in [k]} \left( \hat{\theta}_\lambda(z^n) - \hat{\theta}_\lambda(z^{n\setminus i}) \right)^\top \left( \frac{\partial}{\partial \theta_k} \nabla_{\theta} u_{n-1}(z_j; \zeta^{i,j,1}_{\lambda,n}(z^n), \lambda) \right) \left( \hat{\theta}_\lambda(z^n) - \hat{\theta}_\lambda(z^{n\setminus i}) \right) e_k, \]
where \( e_k \) is \( k \)-th standard unit vector, and such that for all \( \kappa \in [k] \), \( \zeta^{i,j,1}_{\lambda,n}(z^n) = \alpha^{i,j,1}_{\kappa} \hat{\theta}_\lambda(z^n) + (1 - \alpha^{i,j,1}_{\kappa}) \hat{\theta}_\lambda(z^{n\setminus i}) \) for some \( 0 \leq \alpha^{i,j,1}_{\kappa} \leq 1 \). Further, \( \varepsilon^{(i),2}_{\lambda,n} \) is defined as
\[ \varepsilon^{(i),2}_{\lambda,n} \triangleq \sum_{j \in [n] \setminus i} \sum_{\kappa, \nu \in [k]} \hat{e}_k^\top \left( \hat{\theta}_\lambda(z^n) - \hat{\theta}_\lambda(z^{n\setminus i}) \right) \left( \frac{\partial^2}{\partial \theta_{\kappa} \partial \theta_{\nu}} \nabla_{\theta} u_{n-1}(z_j; \zeta^{i,j,2}_{\lambda,n,\nu}(z^n), \lambda) \right) \left( \hat{\theta}_\lambda(z^n) - \hat{\theta}_\lambda(z^{n\setminus i}) \right) e_k, \]
such that for \( \kappa, \nu \in [k] \), \( \zeta^{i,j,2}_{\lambda,n,\nu}(z^n) = \alpha^{i,j,2}_{\kappa,\nu} \hat{\theta}_\lambda(z^n) + (1 - \alpha^{i,j,2}_{\kappa,\nu}) \hat{\theta}_\lambda(z^{n\setminus i}) \) for some \( 0 \leq \alpha^{i,j,2}_{\kappa,\nu} \leq 1 \). Further, we have\[3\]
\[ \| \hat{\theta}_\lambda(z^n) - \hat{\theta}_\lambda(z^{n\setminus i}) \|_\infty = O_p \left( \frac{1}{n} \right), \]
\[ \| \hat{\theta}_\lambda(z^{n\setminus i}) - \hat{\theta}_\lambda^{(i)}(z^n) \|_\infty = O_p \left( \frac{1}{n^2} \right). \]

See the appendix for the proof. Inspired by Theorem [1] we provide an approximation on the cross validation vector.

\textbf{Definition 6 (approximate cross validation vector)} Let \( ACV_{\lambda,i}(z^n) = \ell \left( z_i; \hat{\theta}_\lambda^{(i)}(z^n) \right) \). We call \( \{ACV_{\lambda,i}(z^n)\}_{i \in [n]} \) the approximate cross validation vector. We further call
\[ \text{ACV}_\lambda(z^n) \triangleq \frac{1}{n} \sum_{i=1}^n ACV_{\lambda,i}(z^n) \]
the approximate cross validation estimator of the out-of-sample loss.

We remark that the definition can be extended to leave-\( q \)-out and \( q \)-fold cross validation by replacing the index \( i \) to an index set \( S \) with \( |S| = q \), comprised of the \( q \) left-out samples in \( [4] \).

The cost of the computation of \( \{\hat{\theta}_\lambda^{(i)}(z^n)\}_{i \in [n]} \) is upper bounded by \( O(np + C(n, p)) \) where \( C(n, p) \) is the computational cost of solving for \( \hat{\theta}_\lambda(z^n) \) in \( [2] \); see \( [14] \). Note that the empirical risk minimization problem posed in \( [3] \) requires time at least \( \Omega(np) \). Hence, the overall cost of computation of \( \{\hat{\theta}_\lambda^{(i)}(z^n)\}_{i \in [n]} \) is dominated by solving \( [2] \). On the other hand, the cost of computing the true cross validation performance by naively solving \( n \) optimization problems \( \{\hat{\theta}_\lambda(z^{n\setminus i})\}_{i \in [n]} \) posed in \( [3] \) would be \( O(nC(n, p)) \) which would necessarily be \( \Omega(n^2p) \) making it impractical for large-scale problems.

\textbf{Corollary 2} The approximate cross validation vector is exact for kernel ridge regression. That is, given that the regularized loss function is quadratic in \( \theta \), we have \( \hat{\theta}_\lambda^{(i)}(z^n) = \hat{\theta}_\lambda(z^{n\setminus i}) \) for all \( i \in [n] \).

\textbf{Proof} We notice that the error term \( \varepsilon^{(i)}_{\lambda,n} \) in \( [6] \) only depends on the third derivative of the loss function in a neighborhood of \( \hat{\theta}_\lambda(z^n) \). Hence, provided that the regularized loss function is quadratic in \( \theta \), \( \varepsilon^{(i)}_{\lambda,n} = 0 \) for all \( i \in [n] \). \( \blacksquare \)

\[ X_n = O_p(a_n) \] implies that \( X_n/a_n \) is stochastically bounded with respect to the density function \( p(\cdot) \).
The fact that the cross validation vector could be obtained for kernel ridge regression in closed form without actually performing cross validation is not new, and the method is known as PRESS. In a sense, the presented approximation could be thought of as an extension of this idea to more general loss and regularizer functions while losing the exactness property. We remark that the idea of ALOOCV is also related to that of the influence functions. In particular, influence functions have been used in [14] to derive an approximation on LOOCV for neural networks with large sample sizes. However, we notice that methods based on influence functions usually underestimate over-fitting making them impractical for model selection. In contrast, we empirically demonstrate the effectiveness of ALOOCV in capturing over-fitting and model selection.

In the case of $\ell_1$ regularizer we assume that the support set of $\hat{\theta}_\lambda(z^n)$ and $\hat{\theta}_\lambda(z^{n\setminus i})$ are the same. Although this would be true for large enough $n$ under Assumption[1] it is not necessarily true for a given sample $z^n$ when sample $i$ is left out. Provided that the support set of $\hat{\theta}_\lambda(z^{n\setminus i})$ is known we use the developed machinery in Theorem[1] on the subset of parameters that are non-zero. Further, we ignore the $\ell_1$ regularizer term in the regularized loss function as it does not contribute to the Hessian locally, and we assume that the regularized loss function is otherwise smooth in the sense of Assumption[1]. In this case, the cost of calculating ALOOCV would scale with $O(np_n \log(1/\epsilon))$ where $p_n$ denotes the number of non-zero coordinates in the solution $\hat{\theta}_\lambda(z^n)$.

We remark that although the nature of guarantees in Theorem[1] are asymptotic, we have experimentally observed that the estimator works really well even for $n$ and $p$ as small as 50 in elastic net regression, logistic regression, and ridge regression. Next, we also provide an asymptotic characterization of the approximate cross validation.

**Lemma 3** Under Assumption[1] we have
\[
\overline{CV}_\lambda(z^n) = \tilde{\mathcal{L}}_{z^n}(\hat{\theta}_\lambda(z^n)) + \tilde{\mathcal{R}}_{z^n}(\hat{\theta}_\lambda(z^n), \lambda) + O_p\left(\frac{1}{n^2}\right)
\]
where
\[
\tilde{\mathcal{R}}_{z^n}(\theta, \lambda) = \frac{1}{n(n-1)} \sum_{i \in [n]} \nabla_\theta^T \ell(z_i; \theta) \left[ \tilde{H}_{z^n\setminus i}(\theta, \lambda) \right]^{-1} \nabla_\theta \ell(z_i; \theta).
\]

Note that in contrast to the ALOOCV (in Theorem[1]), the $O_p(1/n^2)$ error term here depends on the second derivative of the loss function with respect to the parameters, consequently leading to worse performance, and underestimation of over-fitting.

4 **Tuning the Regularization Parameters**

Thus far, we presented an approximate cross validation vector that closely follows the predictions provided by the cross validation vector, while being computationally inexpensive. In this section, we use the approximate cross validation vector to tune the regularization parameters for the optimal out-of-sample performance. We are interested in solving $\min_\lambda \left( \overline{CV}_\lambda(z^n) = \frac{1}{n} \sum_{i=1}^n \ell(z_i; \theta_\lambda(z^{n\setminus i})) \right)$. To this end, we need to calculate the gradient of $\hat{\theta}_\lambda(z^n)$ with respect to $\lambda$, which is given in the following lemma.

**Lemma 4** We have $\nabla_\lambda \hat{\theta}_\lambda(z^n) = -\frac{1}{n} \left[ \tilde{H}_{z^n}(\hat{\theta}_\lambda(z^n), \lambda) \right]^{-1} \nabla_\theta \hat{r}(\hat{\theta}_\lambda(z^n))$.

**Corollary 5** We have $\nabla_\lambda \hat{\theta}_\lambda(z^{n\setminus i}) = -\frac{1}{n-1} \left[ \tilde{H}_{z^n\setminus i}(\hat{\theta}_\lambda(z^{n\setminus i}), \lambda) \right]^{-1} \nabla_\theta \hat{r}(\hat{\theta}_\lambda(z^{n\setminus i}))$.

In order to apply first order optimization methods for minimizing $\overline{CV}_\lambda(z^n)$, we need to compute its gradient with respect to the tuning parameter vector $\lambda$. Applying the simple chain rule implies
\[
\nabla_\lambda \overline{CV}_\lambda(z^n) = \frac{1}{n} \sum_{i=1}^n \nabla_\theta^T \hat{\theta}_\lambda(z^{n\setminus i}) \nabla_\theta \ell(z_i; \hat{\theta}_\lambda(z^{n\setminus i}))
\]
\[
= -\frac{1}{n(n-1)} \sum_{i=1}^n \nabla_\theta \hat{r}(\hat{\theta}_\lambda(z^{n\setminus i})) \left[ \tilde{H}_{z^n\setminus i}(\hat{\theta}_\lambda(z^{n\setminus i})) \right]^{-1} \nabla_\theta \ell(z_i; \hat{\theta}_\lambda(z^{n\setminus i})),
\]
Figure 1: The progression of the loss when Algorithm 1 is applied to ridge regression with diagonal regressors.

Figure 2: The progression of $\lambda$’s when Algorithm 1 is applied to ridge regression with diagonal regressors.

where (15) follows by substituting $\nabla \hat{\theta}^i (z^n)$ from Corollary 5. However, (15) is computationally expensive and almost impossible in practice even for medium size datasets. Hence, we use the ALOOCV from (4) (Theorem 1) in (14) to approximate the gradient.

Let

$$g^{(i)} (z^n) \triangleq -\frac{1}{n-1} \nabla_{\theta} r \left( \hat{\theta}^{(i)} (z^n) \right) \left[ \hat{H}_{z^n} \left( \hat{\theta}^{(i)} (z^n) \right) \right]^{-1} \nabla_{\theta} \ell \left( z^n; \hat{\theta}^{(i)} (z^n) \right).$$

Further, motivated by the suggested ALOOCV, let us define the approximate gradient $g^{(i)} (z^n)$ as $g^{(i)} (z^n) \triangleq \frac{1}{n} \sum_{i \in [n]} g^{(i)} (z^n)$. Based on our numerical experiments, this approximate gradient closely follows the gradient of the cross validation, i.e., $\nabla \hat{\lambda} CV (z^n) \approx g^{(i)} (z^n)$. Note that this approximation is straightforward to compute. Therefore, using this approximation, we can apply the first order optimization algorithm 1 to optimize the tuning parameter $\lambda$.

**Algorithm 1** Approximate gradient descent algorithm for tuning $\lambda$

```
Initialize the tuning parameter $\lambda^0$, choose a step-size selection rule, and set $t = 0$

for $t = 0, 1, 2, \ldots$ do
    calculate the approximate gradient $g^{(i)} (z^n)$
    set $\lambda^{t+1} = \lambda^t - \alpha^t g^{(i)} (z^n)$
end for
```

more computationally efficient compared to LOOCV (saving a factor of $n$), it might still be computationally expensive for large values of $n$ as it still scales linearly with $n$. Hence, we also present an online version of the algorithm using the stochastic gradient descent idea; see Algorithm 2.

**Algorithm 2** Stochastic (online) approximate gradient descent algorithm for tuning $\lambda$

```
Initialize the tuning parameter $\lambda^0$ and set $t = 0$

for $t = 0, 1, 2, \ldots$ do
    choose a random index $i_t \in \{1, \ldots, n\}$
    calculate the stochastic gradient $g^{(i_t)} (z^n)\triangleq(16)$
    set $\lambda^{t+1} = \lambda^t - \alpha^t g^{(i_t)} (z^n)$
end for
```

5 Numerical Experiments

Ridge regression with diagonal regressors: We consider the following regularized loss function:

$$w_n (z; \theta, \lambda) = \ell (z; \theta) + \frac{1}{n} \lambda^T \Sigma (\theta) = \frac{1}{2} (y - \theta^T x)^2 + \frac{1}{2n} \lambda^T \Sigma (\lambda) \theta.$$
In other words, we consider one regularization parameter per each model parameter. To validate the proposed optimization algorithm, we consider a scenario with $p = 50$ where $x$ is drawn i.i.d. from $\mathcal{N}(0, I_p)$. We let $y = \theta^\top x + \epsilon$ where $\theta_1 = \ldots = \theta_{40} = 0$ and $\theta_{41}, \ldots, \theta_{50} \sim \mathcal{N}(0, 1)$ i.i.d. and $\epsilon \sim \mathcal{N}(0, 1)$. We draw $n = 150$ samples from this model, and apply Algorithm 1 to optimize for $\lambda = (\lambda_1, \ldots, \lambda_{50})$. The problem is designed in such a way that out of 50 features, the first 40 are irrelevant while the last 10 are important. We initialize the algorithm with $\lambda_1 = \ldots = \lambda_{40} = 1/3$ and compute ACV using Theorem 1. Recall that in this case, ACV is exactly equivalent to CV (see Corollary 2). Figure 1 plots ALOOCV, the out-of-sample loss, and the mean value of $\lambda$ calculated over the irrelevant and relevant features respectively. As expected, the $\lambda$ for an irrelevant feature is set to a larger number, on the average, compared to that of a relevant feature. Finally, we remark that the optimization of 50 tuning parameters in 800 iterations took a mere 28 seconds on a PC.
Logistic regression: The second example that we consider is logistic regression:

$$w_n(z; \theta, \lambda) = \ell(z; \theta) + \frac{1}{n} \lambda^\top r(\theta) = H(y || \text{sigmoid}(\theta_0 + \theta^\top x)) + \frac{1}{2n} \lambda ||\theta||^2.$$  

where $H(\cdot | \cdot)$ for any $u \in [0,1]$ and $v \in (0,1)$ is given by $H(u|v) := u \log \frac{1}{v} + (1 - u) \log \frac{1}{1-v}$, and denotes the binary cross entropy function, and $\text{sigmoid}(x) := 1/(1 + e^{-x})$ denotes the sigmoid function. In this case, we only consider a single regularization parameter. Since the loss and regularizer are smooth, we resort to Theorem 1 to compute ACV. We applied logistic regression on MNIST and CIFAR-10 image datasets where we used each pixel in the image as a feature according to the aforementioned loss function. In MNIST, we classify the digits 2 and 3 while in CIFAR-10, MNIST and CIFAR-10 image datasets where we used each pixel in the image as a feature according to the sigmoid function. In this case, we only consider a single regularization parameter. Since the loss and regularizer are smooth, we resort to Theorem 1 to compute ACV. We applied logistic regression on MNIST and CIFAR-10 image datasets where we used each pixel in the image as a feature according to the sigmoid function.

Elastic net regression: Finally, we consider the popular elastic net regression problem [31]:

$$w_n(z; \theta, \lambda) = \ell(z; \theta) + \frac{1}{n} \lambda^\top r(\theta) = \frac{1}{2} (y - \theta^\top x)^2 + \frac{1}{n} \lambda_1 ||\theta||_1 + \frac{1}{2n} \lambda_2 ||\theta||^2.$$  

In this case, there are only two regularization parameters to be optimized for the quasi-smooth regularized loss. Similar to the previous case, we consider $y = \theta^\top x + \epsilon$ where $\theta_{\epsilon} = \kappa \rho_{\epsilon} \psi_{\epsilon}$ where $\rho_{\epsilon}$ is a Bernoulli(1/2) RV and $\psi_{\epsilon} \sim \mathcal{N}(0,1)$. Hence, the features are weighted non-uniformly in $y$ and half of them are zeroed out on the average. We apply both Algorithms 1 and 2 where we used the approximation in Theorem 1 and the explanation on how to handle $\ell_1$ regularizers to compute ACV. We initialized with $\lambda_1 = \lambda_2 = 0$. As can be seen on the left panel (Figure 4), ACV closely follows CV in this case. Further, we see that both algorithms are capable of significantly reducing the loss after only a few iterations. The right panel compares the run-time of the algorithms vs. the number of samples. This confirms our analysis that the run-time of CV scales quadratically with $O(n^2)$ as opposed to $O(n)$ in ACV. This impact is more signified in the inner panel where the run-time ratio is plotted.

Acknowledgement

This work was supported in part by DARPA under Grant No. W911NF-16-1-0561. The authors are thankful to Jason D. Lee (USC) who brought to their attention the recent work [14] on influence functions for approximating leave-one-out cross validation.
References


