Online Hyperparameter Search Interleaved with Proximal Parameter Updates

Luis M. Lopez-Ramos, *Member, IEEE,* and Baltasar Beferull-Lozano, *Senior Member, IEEE*

***Abstract*—There is a clear need for efficient hyperparameter optimization (HO) algorithms for statistical learning, since commonly applied search methods (such as grid search with N-fold cross-validation) are inefficient and/or approximate. Previously existing gradient-based HO algorithms that rely on the smoothness of the cost function cannot be applied in problems such as Lasso regression. In this contribution, we develop a HO method that relies on the structure of proximal gradient methods and does not require a smooth cost function. Such a method is applied to Leave-one-out (LOO)-validated Lasso and Group Lasso, and an online variant is proposed. Numerical experiments corroborate the convergence of the proposed methods to stationary points of the LOO validation error curve, and the improved efficiency and stability of the online algorithm.**

***Index Terms*—Hyperparameter optimization, regression, online learning, proximal gradient descent.**

I. Introduction

Given their proven utility to control the model complexity, hyperparameters are crucial for a successful application of statistical learning schemes in many engineering problems. The generalization capability and performance of such schemes on unknown instances can be improved with a careful hyperparameter selection. Regularized models control the trade-off between a data fidelity term and a complexity term known as regularizer by means of one or several hyperparameters. Sparsity-promoting, regularized models such as Lasso, Group Lasso, and Elastic-Net can get their regression weights optimized efficiently via proximal gradient descent (PGD) and its variants. The sparsity of their estimates depends on a hyperparameter, and the associated hyperparameter optimization (HO) is a non-convex, challenging problem [1].

Given a dataset in batch form, a commonly applied criterion for HO is the leave-one-out (LOO) validation error, because it reflects the ability of an estimator to predict outputs for unobserved patterns [2]. The computational cost of evaluating the LOO validation error grows superlinearly with the number of data points, so that it is often approximated by N-fold cross validation (CV) with a small N (e.g., 10). Although there are many HO algorithms available (see [3, Ch. 1] for a recent review of the state of the art), it is still an extended practice to search for (sub)optimal hyperparameters using grid search or random search [1], [4], because of their simplicity.

Improved random search algorithms, termed *configuration-evaluation* methods, focus computation resources in promising hyperparameter configurations by quickly eliminating poor ones, important examples being Hyperband [5] and Bayesian optimization-based approaches in [6]. Many of these are black-box methods that ignore the structure of the HO problem.

On the other hand, gradient-based (exact and approximate) HO methods have been proposed recently for problems where the cost function is smooth. Several recent approaches formulate a bi-level program where an inner program is the optimization of the model parameters (model weights in the case of regression) and the outer program is the minimization of a surrogate of the generalization capability (e.g. validation MSE). In particular, [7] applies the implicit function Theorem to a stationarity condition to obtain the hypergradient (gradient of the outer cost function w.r.t. the hyperparameters); however, this approach requires calculating the Hessian w.r.t. the model parameters and, consequently, it cannot be applied to widely used non-smooth regularizers (such as Lasso/Group Lasso).

The approaches in [8]–[10] obtain a hypergradient by modeling the regression weight optimization as a dynamical system, where the state space is the parameter space and each iteration corresponds to a mapping from/to the same space. While [8] requires such a mapping to be invertible, [9], [10] avoid that requirement by resorting to an approximation. This work combines ideas from [7], [9] to formulate a different implicit equation, derive the exact hypergradient, and develop a method that can work with non-smooth regularizers and, additionally, admits an online variant.

In the more challenging setting of processing data in streaming with a time-varying distribution, algorithms that adaptively compute the regularization parameter are in order. One existing adaptive approach [8] does so for different time segments or data windows, and is specific for Lasso. On the contrary, the approach in the present paper is general enough to be applied to several generalizations of Lasso, such as Group Lasso.

Approaches to HO based on neural networks (NN) exist, such as [11] which predicts optimal regression weights for a hyperparameter vector, together with approximations that alternate NN weight updates with hyperparameter updates. Sparse recovery algorithms which model an iterative soft-thresholding algorithm (ISTA) as a deep network (see [12] and references therein) can be understood as "learning" a sequence of hyperparameters. However, it is not clear how these approaches can be adapted to process data in streaming.

Another approximation alleviating computation in HO [13] exploits the structure of specific estimators such as Lasso to directly compute an approximation of the LOO error metric at a low cost. Despite the reduced computation, using this ap-

This work was supported by grants SFI Offshore Mechatronics 237896/E30, PETROMAKS Smart-Rig 244205/E30, IKTPLUSS INDURB 270730/O70

The authors are with the WISENET Center, Dept. of ICT, University of Agder, Jon Lilletunsvei 3, Grimstad, 4879 Norway. E-mails:{luismiguel.lopez, baltasar.beferull}@uia.no.

proximation for HO still requires a black-box search scheme, which does not scale well with the dimensionality.

In this paper, we propose and evaluate a method that jointly optimizes the weights and hyperparameters a model with a proximable, non-smooth regularizer; and converges to a stationary point of the LOO error curve. We explicitly apply our approach to Lasso and Group Lasso. The formulation is inspired by the forward-mode gradient computation in [9], but where we use efficient approximations based on online (stochastic) gradient descent.

The contributions and structure of the present paper are listed in the following: Sec. II provides the general formulation for the HO in supervised learning and presents the use of PGD for our problem. In Sec. III, we present the derivation of the hypergradient (gradient w.r.t the hyperparameters). In Sec. IV, our method is applied to estimators with non-smooth cost functions, in particular Lasso and Group Lasso. The main contributions are presented in Sec. V, consisting in an online algorithm and an approximate scheme, both aimed at saving computation. Sec. VI contains numerical tests with synthetic data, and concludes the paper.

II. Problem formulation

Given a set of training input/label pairs $\{x_i, y_i\}_{i=1}^N$, with $x_i \in \mathbb{R}^P$ and $y_i \in \mathbb{R}$, consider the problem of minimizing a linear combination of empirical risk (data fit) and structural risk (regularization term):

$$w^*(\lambda, \mathcal{B}) := \arg\min_w \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \ell_i(w) + \lambda^\top \Omega(w), \qquad (1)$$

for $\lambda \in \mathbb{R}_+^D$. This can be for instance particularized to the Lasso regression problem with $w \in \mathbb{R}^P$, $\ell_i(w) = (y_i - x_i^\top w)^2$, and $\Omega(w) = \|w\|_1$; section IV discusses other estimators.

It is well known that a low empirical risk does not guarantee good generalization. The role of regularization is to select the right model complexity to avoid overfitting, and the right choice of the hyperparameter λ is crucial. To this end, any estimator in the form (1) can be embedded in the bi-level optimization problem (minimization of the validation error):

$$\breve{\lambda}^* := \arg\min_\lambda \frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \ell_j^{\mathrm{VAL}}(w^*(\lambda, \mathcal{B}_j)). \qquad (2)$$

where $\mathcal{V}$ denotes the set of validation samples, and $\mathcal{B}_j$ denotes the training batch associated with the j-th validation sample. Among the possible choices for the validation error metric [3], $\ell_j^{\mathrm{VAL}} = (y_j - x_j^\top w)^2$ is appropriate for regression. Since (2) may have several local minima, the notation $\breve{\lambda}^*$ is reserved for a global minimizer, whereas λ^* will be used throughout the text to denote a stationary point.

Regarding the collection of training batches and the validation samples: In holdout validation, $\mathcal{B}_j = \mathcal{B}\ \forall j$, and $\mathcal{V} \cap \mathcal{B} = \emptyset$. In N-fold cross-validation (CV), $\mathcal{V}$ is the train-and-validate dataset; the *folds* $\{\mathcal{F}_1, \ldots, \mathcal{F}_N\}$ are a partition of $\mathcal{V}$; and $\mathcal{B}_j = \bigcup_{j \notin \mathcal{F}_n} \mathcal{F}_n$. LOO is a special case of CV where $N = |\mathcal{V}|$, and $\mathcal{F}_i = \{i\}\ \forall\ i$; and therefore, $\mathcal{B}_j = \mathcal{V} \setminus \{j\}$.

The rest of this section reviews how $w^*(\lambda, \mathcal{B}_j)$ is obtained. The next section will discuss the minimization of (2) via the computation of the gradient w.r.t. the hyperparameter λ, also referred to as *hyper-gradient* [9], [14].

A. Proximal Gradient Descent

The proximal gradient descent (PGD) algorithm allows to iteratively compute $w^*(\lambda, \mathcal{B}_j)$ given the training batch $\mathcal{B}_j$ and the hyperparameter λ, and it is advocated here for its simplicity. Extending our formulation to accommodate algorithms such as the accelerated PGD (which gives rise to FISTA [15] when applied to ℓ_1-regularized problems) is out of the scope of the present paper and left as future work.

Given a function Ψ, the proximity (prox) operator is defined as [16]

$$\mathrm{prox}_\Psi^\eta(\boldsymbol{v}) \triangleq \underset{\boldsymbol{x} \in \mathrm{dom}\ \Psi}{\arg\min} \left[\Psi(\boldsymbol{x}) + \frac{1}{2\eta} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right]. \qquad (3)$$

If Ω is such that its prox operator admits closed form, it is said that Ω is a *proximable* function, and problem (1) can be solved efficiently via proximal gradient descent (PGD):

$$w^{(k+1)} = \mathrm{prox}_\Omega^{\lambda\alpha^{(k)}} \left(w^{(k)} - \frac{\alpha^{(k)}}{|\mathcal{B}_j|} \sum_{i \in \mathcal{B}_j} \left(\nabla_w \ell_i(w^{(k)}) \right) \right) \qquad (4)$$

where $\alpha^{(k)}$ is a step size sequence satisfying $\alpha^{(k)} < 1/L$, where L is the Lipschitz smoothness parameter of the empirical risk (aggregate loss component of the cost function). In fact, for $\alpha^{(k)} < 1/L$, it holds that $w_j^{(k)} \xrightarrow[k \to \infty]{} w^*(\lambda, \mathcal{B}_j)$. The PGD step (4) is the composition of a gradient step with the prox operator, and the iteration is frequently split in two steps, yielding the equivalent *forward-backward* iterations:

$$w_f^{(k)} = F_{\mathcal{B}}^{\alpha^{(k)}}(w^{(k)}) \triangleq w^{(k)} - \frac{\alpha^{(k)}}{|\mathcal{B}_j|} \sum_{i \in \mathcal{B}} \nabla_w \ell_i(w^{(k)}) \qquad (5a)$$

$$w^{(k+1)} = \mathrm{prox}_\Omega^{\lambda\alpha^{(k)}}(w_f^{(k)}) \qquad (5b)$$

Moreover, for $\alpha \in (0, 1/L]$ the optimality condition holds:

$$w^*(\lambda, \mathcal{B}) = \mathrm{prox}_\Omega^{\lambda\alpha}(F_{\mathcal{B}}^\alpha(w^*(\lambda, \mathcal{B}))). \qquad (6)$$

III. Computing the Hyper-gradient

The condition in (6) establishes optimality w.r.t. the weight vector, but not w.r.t. the hyperparameter λ. To optimize over λ, we leverage the *forward-mode* gradient computation described by [9] in this section. The condition for λ^* being a stationary point for the optimization in (2) is:

$$\sum_{j \in \mathcal{V}} \nabla_\lambda \ell_j^{\mathrm{VAL}}(w^*(\lambda^*, \mathcal{B}_j)) = 0. \qquad (7)$$

The hyper-gradient can be written using the chain rule as

$$\nabla_\lambda \ell_j^{\mathrm{VAL}}(w^*(\lambda, \mathcal{B})) = \left(\frac{\partial w^*(\lambda, \mathcal{B})}{\partial \lambda} \right)^\top \nabla_w \ell_j^{\mathrm{VAL}}(w^*(\lambda, \mathcal{B})), \qquad (8)$$

where the argument of $^\top$ is the derivative (Jacobian) matrix (column vector if λ is scalar). In the sequel, we leverage the technique in [9] to compute the latter.

Consider a generic iterative algorithm, whose t-th iterate is $s_t \in \mathbb{R}^P$, and a hyperparameter vector $\lambda \in \mathbb{R}^D_+$. The t-th iteration can be expressed as: $s_t = \mathcal{M}_t(s_{t-1}, \lambda)$, where

$$\mathcal{M}_t : (\mathbb{R}^P \times \mathbb{R}^D) \to \mathbb{R}^P$$

is a smooth mapping that represents the operation performed at the latter. The following equation [9, eq. (13)] is fulfilled by the iterates s_t:

$$\frac{ds_t}{d\lambda} = \frac{\partial \mathcal{M}_t(s_{t-1}, \lambda)}{\partial s_{t-1}} \frac{ds_{t-1}}{d\lambda} + \frac{\partial \mathcal{M}_t(s_{t-1}, \lambda)}{\partial \lambda} \tag{9}$$

In the case of PGD, the mapping $\mathcal{M}_k$ is the composition $\operatorname{prox}_\Omega^{\lambda\alpha(k)} \circ F_\mathcal{B}^{\alpha(k)}$ [cf. (5)]. For simplicity, we will consider in the sequel a constant step size $\alpha^{(k)} = \alpha$ for PGD, so that $\mathcal{M}_k = \mathcal{M} = \operatorname{prox}_\Omega^{\lambda\alpha} \circ F_\mathcal{B}^\alpha$, and

$$\frac{dw^{(k+1)}}{d\lambda} = A(w_f^{(k)}) \frac{\partial F_\mathcal{B}^\alpha(w^{(k)})}{\partial w^{(k)}} \frac{dw^{(k)}}{d\lambda} + B(w_f^{(k)}) \tag{10}$$

$$\text{where } A(w_f) \triangleq \frac{\partial(\operatorname{prox}_\Omega^{\lambda\alpha})(w_f)}{\partial w_f}, B(w_f) \triangleq \frac{\partial(\operatorname{prox}_\Omega^{\lambda\alpha})(w_f)}{\partial \lambda}. \tag{11}$$

The derivations so far have followed a path common to [10], where an approximation to the hypergradient is computed by reverse-mode gradient computation [9]. However, differently to that work, in our approach we identify a fixed point equation for the derivatives at the convergence point of PGD:

$$\frac{dw^*(\lambda,\mathcal{B})}{d\lambda} = A(w_f^*) \frac{\partial F_\mathcal{B}^\alpha(w^*(\lambda,\mathcal{B}))}{\partial w^*(\lambda,\mathcal{B})} \frac{dw^*(\lambda,\mathcal{B})}{d\lambda} + B(w_f^*) \tag{12}$$

where $w_f^* \triangleq F_\mathcal{B}^\alpha(w^*(\lambda,\mathcal{B}))$; if the linear equation has a solution, it can be expressed in closed form as $\frac{dw^*(\lambda,\mathcal{B})}{d\lambda} = Z_\mathcal{B}(w^*(\lambda,\mathcal{B}))$, where

$$Z_\mathcal{B}(w^*(\lambda,\mathcal{B})) \triangleq \left(\boldsymbol{I} - A(w_f^*) \frac{\partial F_\mathcal{B}^\alpha(w^*(\lambda,\mathcal{B}))}{\partial w^*(\lambda,\mathcal{B})} \right)^{-1} B(w_f^*). \tag{13}$$

A. Hyper-gradient descent (HGD)

If the iterates

$$\lambda^{(k+1)} := \Big[\lambda^{(k)} - \frac{\beta^{(k)}}{|\mathcal{V}|} \times \sum_{j\in\mathcal{V}} \Big(Z_\mathcal{B}(w^*(\lambda^{(k)}, \mathcal{B}_j)) \Big)^\top \nabla_w \ell_j^{\text{VAL}}(w^*(\lambda^{(k)}, \mathcal{B}_j)) \Big]_+ \tag{14}$$

(where $[\cdot]_+$ denotes projection onto the positive orthant) are executed, with an appropriate step size sequence $\beta^{(k)}$, the sequence $\lambda^{(k)}$ will converge to a stationary point of (2). The computational complexity of (14) is $\mathcal{O}(P^3)$ per iteration because it is dominated by the matrix inversion in (13).

Remark. Existence of $Z_\mathcal{B}(\cdot)$ requires the prox operator to be smooth, but important estimators (e.g. Lasso) involve non-smooth prox operators. In the next section, a slight modification of HGD is proposed to deal with those problems.

IV. Non-smooth prox operators

For estimators which regularizing function is nonsmooth frequently the prox operator is also nonsmooth. At some points, the derivatives of such prox operators will not exist, and thus $Z_j(w^*(\lambda,\mathcal{B}_j))$ may not be computable. One can instead compute a valid subderivative (which will be denoted by $\tilde{Z}_j(w^*(\lambda,\mathcal{B}_j))$) by replacing the derivatives of the prox operator with the corresponding subderivatives.

If $Z_j(w^*(\lambda,\mathcal{B}_j))$ is replaced in (14) with $\tilde{Z}_j(w^*(\lambda,\mathcal{B}_j))$, the resulting algorithm will be termed hereafter as *hyper-subgradient descent (HSGD)*. The HSGD is advocated in this section to perform HO for estimation problems trainable via PGD (where the loss is differentiable and the regularization function is nonsmooth), and it is explicitly derived for Lasso and Group Lasso. Some of the functions that have been presented before generically will be particularized to facilitate the readability of the derivations and algorithms.

Regularized least-squares (LS) linear estimators such as Lasso use the loss function $\ell_i(w) = (y_i - x_i^\top w)^2$. Consequently, the forward operator and its Jacobian are

$$F_{\mathcal{B}_j}^\alpha(w) = w - \alpha(\boldsymbol{\Phi}_j w - r_j), \quad \text{and} \quad \frac{\partial F_{\mathcal{B}_j}^\alpha(w)}{\partial w} = (\boldsymbol{I} - \alpha\boldsymbol{\Phi}_j),$$

where $\boldsymbol{\Phi}_j := \frac{1}{|\mathcal{B}_j|} \sum_{i\in\mathcal{B}_j} x_i x_i^\top$, and $r_j := \frac{1}{|\mathcal{B}_j|} \sum_{i\in\mathcal{B}_j} y_i x_i$. If the LOO validation scheme is chosen, then $\boldsymbol{\Phi}_j$ can be computed efficiently as

$$\boldsymbol{\Phi}_j := \tfrac{1}{N-1}(N\boldsymbol{\Phi} - x_i x_i^\top), \quad r_j := \tfrac{1}{N-1}(Nr - x_i y_i); \tag{15}$$

$$\text{with } \boldsymbol{\Phi} \triangleq \tfrac{1}{N} \textstyle\sum_{i\in\mathcal{V}} x_i x_i^\top, \quad r \triangleq \tfrac{1}{N} \textstyle\sum_{i\in\mathcal{V}} x_i y_i. \tag{16}$$

If the validation metric is $\ell_j^{\text{VAL}} = (y_j - x_j^\top w)^2$, its gradient is $\nabla_w \ell_j^{\text{VAL}}(w) = x_j(x_j^\top w - y_j)$. The equations for particular cases of $\Omega(\cdot)$ will be presented after the HSGD algorithm.

A. Hyper-subgradient descent (HSGD)

Let $\tilde{A}_j(w_f)$ and $\tilde{B}_j(w_f)$ be valid subderivative (sub-Jacobian) matrices of $\operatorname{prox}_\Omega^{\lambda\alpha}(w_f)$ w.r.t. w_f and λ, respectively. Then, a valid subderivative matrix of $w^*(\lambda,\mathcal{B}_j)$ with respect to λ is [cf. (13)]

$$\tilde{Z}_j(w^*(\lambda,\mathcal{B}_j)) := \Big(\boldsymbol{I} - \tilde{A}_j(w_f^*)(\boldsymbol{I} - \alpha\boldsymbol{\Phi}_j) \Big)^{-1} \tilde{B}_j(w_f^*); \tag{17}$$

where $w_f^* := F_\mathcal{B}^\alpha(w^*(\lambda,\mathcal{B}))$; and the HSGD iterates can be written as

$$\lambda^{(k+1)} := \Big[\lambda^{(k)} - \beta^{(k)} \times \sum_{j\in\mathcal{V}} \Big(\tilde{Z}_j(w^*(\lambda^{(k)}, \mathcal{B}_j)) \Big)^\top x_j(x_j^\top w^*(\lambda^{(k)}, \mathcal{B}_j) - y_j) \Big]_+ \tag{18}$$

The per-iteration complexity is $\mathcal{O}(P^3)$ [cf. (14)].

Remark. If $\boldsymbol{\Phi}_j$, then the inverse at (17) exists, but this is not guaranteed if $\boldsymbol{\Phi}_j$ is rank defficient (which happens when $P < N+1$, and may also happen when the input data x_j have a high degree of collinearity). In such cases, the LS solution of the linear system can be used.

B. Application of HSGD to Lasso and Group Lasso

Diverse choices of Ω give rise to different regularized estimators, associated prox operators, and HSGD iterates.

1) Lasso: The regularizer is $\Omega(w) = \|w\|_1$; its prox operator is known as soft-thresholding $S_{\alpha\lambda}(w) \triangleq \text{prox}^{\alpha\lambda}_{\|\cdot\|_1}(w)$ [17], and the latter can be computed entrywise as

$$[S_{\alpha\lambda}(w_f)]_n := [w_f]_n \left[1 - \frac{\alpha\lambda}{|[w_f]_n|}\right]_+ . \tag{19}$$

The corresponding subderivatives $\tilde{A}(w_f) \in \mathbb{R}^{P\times P}$, and $\tilde{B}(w_f) \in \mathbb{R}^{P\times 1}$ are defined so that $\tilde{A}(w_f)$ is diagonal and

$$[\tilde{A}(w_f)]_{nn} = \mathbb{1}\{|[w_f]_n| \geq \alpha\lambda\} \tag{20a}$$

$$[\tilde{B}(w_f)]_n = \alpha\left(\mathbb{1}\{[w_f]_n \leq -\alpha\lambda\} - \mathbb{1}\{[w_f]_n \geq \alpha\lambda\}\right). \tag{20b}$$

2) Group Lasso: The regularizer depends on an a priori defined group structure. With N_g denoting the number of groups, let $\{\mathcal{K}_1, \mathcal{K}_2, ...\mathcal{K}_{N_g}\}$ be a partition of $\{1, 2, ..., P\}$ (recall that $w \in \mathbb{R}^P$). Let $[w]_{\mathcal{K}}$ denote the sub-vector of w containing the components indexed by $\mathcal{K}$. The regularizer is $\Omega(w) = \|w\|_{2,1} \triangleq \sum_{g=1}^{N_g} \|w_{\mathcal{K}_g}\|_2$; its prox op. is known as multidimensional soft-thresholding $S^G_{\alpha\lambda}(w) \triangleq \text{prox}^{\alpha\lambda}_{\|\cdot\|_{2,1}}(w)$ [18], and the latter can be computed group-wise as

$$[S^G_{\alpha\lambda}(w_f)]_{\mathcal{K}} = [w_f]_{\mathcal{K}} \left[1 - \frac{\alpha\lambda}{\|[w_f]_{\mathcal{K}}\|_2}\right]_+ . \tag{21}$$

With $\mathcal{K}(n)$ denoting the subset of the partition where n belongs, the corresponding subderivative matrices $\tilde{A}(w_f) \in \mathbb{R}^{P\times P}$, and $\tilde{B}(w_f) \in \mathbb{R}^{P\times 1}$ are defined so that $\tilde{A}(w_f)$ is diagonal, and

$$[\tilde{A}(w_f)]_{nn} = \mathbb{1}\{\|[w_f]_{\mathcal{K}(n)}\|_2 \geq \alpha\lambda\} \tag{22a}$$

$$[\tilde{B}(w_f)]_n = \begin{cases} -\alpha\frac{[w_f]_{\mathcal{K}(n)}}{\|[w_f]_{\mathcal{K}(n)}\|_2}, & \|[w_f]_{\mathcal{K}(n)}\|_2 \geq \alpha\lambda, \\ 0, & \|[w_f]_{\mathcal{K}(n)}\|_2 < \alpha\lambda. \end{cases} \tag{22b}$$

Algorithm 1 Hyper-subgradient descent for Lasso or Group Lasso

Input: $\{x_i, y_i\}_{i=1}^N$, $\{\beta^{(k)}\}_k$, $\lambda^{(1)}$
Output: λ^*

Compute $\mathbf{\Phi}$, r via (16)
$\alpha = 1/\rho(\mathbf{\Phi})$
for $k = 1, 2, \ldots$ **do** (until convergence)
  for $j = 1, \ldots, N$ **do**
    Compute $\mathbf{\Phi}_j$, r_j via (15)
    for $m = 1, 2, \ldots$ **do** (until convergence) ▷ PGD
      $w_f^{(m)} = w^{(m-1)} - \alpha(\mathbf{\Phi}_j w^{(m-1)} - r_j)$
      Compute $w^{(m)}$ via (19) or (21)
    Compute $\tilde{A}_j(w_f^*), \tilde{B}_j(w_f^*)$ via (20) or (22)
    Compute $\tilde{Z}_j(w^*(\lambda^{(k)}, \mathcal{B}_j))$ via (17)
  Update $\lambda^{(k+1)}$ via (18)

The HSGD algorithm applied to Lasso and Group Lasso is summarized in the **Algorithm** 1.

Remark. The approach here is flexible enough to be extended to accomodate any $\Omega(\cdot)$ as long as it is convex and proximable. This enables extensions to other estimators.

V. Approximate Algorithms

This section presents two approximations that improve the efficiency of HSGD.

A. Online Hyper-subgradient Descent (OHSGD)

To avoid having to evaluate $w^*(\lambda, \mathcal{B}_j) \forall j$ in each iteration of HSGD, the online optimization technique is applied, consisting in one gradient descent iteration per j, using the corresponding contribution to the subgradient (stochastic subgradient):

$$j(k) := k \mod |\mathcal{V}| \tag{23a}$$

$$w^{(k)} := w^*(\lambda^{(k)}, \mathcal{B}_{j(k)}) \tag{23b}$$

$$\lambda^{(k+1)} := \Big[\lambda^{(k)} - \beta^{(k)} \times \left(\tilde{Z}_{j(k)}(w^{(k)})\right)^\top x_{j(k)}(x_{j(k)}^\top w^{(k)} - y_{j(k)})\Big]_+ \tag{23c}$$

To save computation, the instance of PGD that calculates $w^*(\lambda^{(k)}, \mathcal{B}_{j(k)})$ should be initialized at $w^{(k-|\mathcal{V}|)}$ if $k > |\mathcal{V}|$.

B. OHSGD with inexact weight vector

The algorithm (23) requires to evaluate $w^*(\lambda^{(k)}, \mathcal{B}_{j(k)})$. The iterates produced by PGD converge to the exact optimizer, but in practice one has to stop the inner loop after a certain stopping criterion is met. Let $m(k)$ denote the number of iterations in the k-th (inner) loop; clearly, $\|w_j^{(m(k))} - w^*(\lambda, \mathcal{B}_{j(k)})\|$ decreases with $m(k)$. Evaluating $\tilde{Z}(\cdot)$ for a coarse approximation of $w^*(\lambda, \mathcal{B}_{j(k)})$ produces an approximate (stochastic) hyper-subgradient. A coarse approximation of the hyper-subgradient is usually enough for OHSGD updates to bring $\lambda^{(k)}$ closer to λ^*, allowing to run fewer iterations of PGD at each OHSGD iteration, which further reduces the amount of computation.

VI. Numerical Results

An experiment has been run to visualize the convergence of HSGD and OHSGD to stationary points of the LOO error curve. Synthetic data have been generated with i.i.d. input vectors $x_i \in \mathbb{R}^{200}$ from a standard Gaussian distribution, and $y_i := w_{\text{true}}^\top x_i + \epsilon_i$, with w_{true} being a 10-sparse vector, and ϵ_i generated i.i.d. so that y_i has a signal-to-noise ratio (SNR) of 0.3. The train-and-validate set contains 400 samples.

The proposed algorithms HSGD and OHSGD were run with different constant stepsizes $\beta^{(k)} = \beta$. The values of the λ iterates are shown in Fig. 1 in terms of the number of times the matrix inversion in (17) is executed. The normalized mean square error (NMSE) evaluated via LOO for a grid of uniformly sampled values of λ is shown at the right; the λ axis of both figures are aligned to emphasize the convergence to (neigborhoods of) stationary points.

Two main observations about Fig. 1: a) OHSGD requires fewer evaluations of (17) to approach local minima, and b) OHSGD is less sensitive than HSGD to the choice of β, (HSGD failed in this experiment for the two larger values of

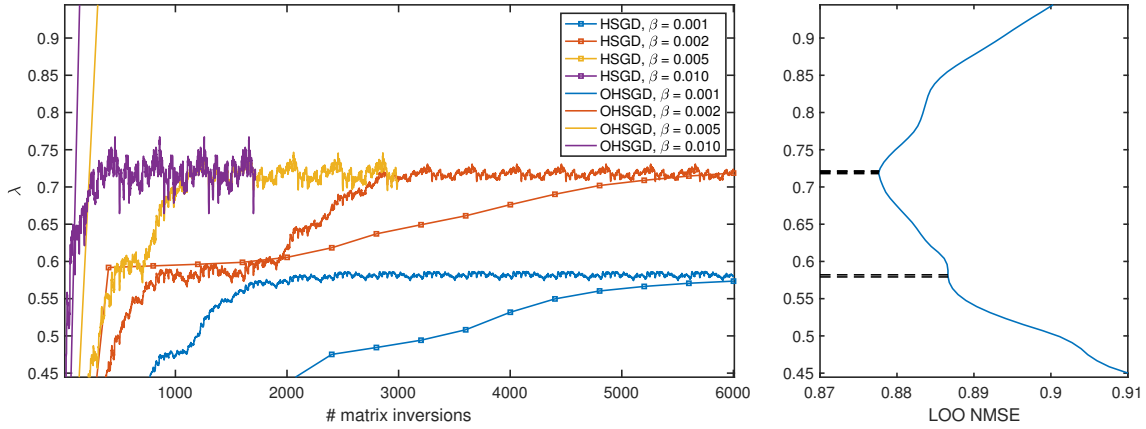

Fig. 1: Left: Iterates of HSGD and OHSGD for different values of β vs. number of times (17) is executed. Right: LOO error curve, with dashed horizontal lines marking convergence values of the HSGD iterates. Note that the λ-axis of both figures are aligned.

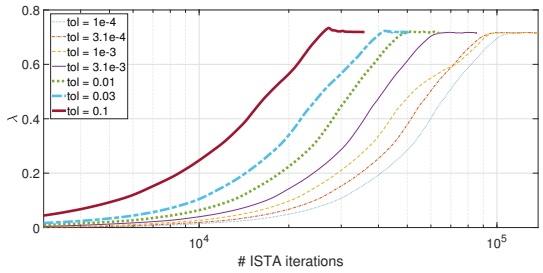

Fig. 2: OHSGD iterates for $\beta = 6e-5$, and different values of the tolerance to stop PGD/ISTA.

β). The risk of converging to low-performance local minima of the non-convex LOO curve can be mitigated in two ways: a) using diverse intial values of λ, and b) adding a momentum term to the OHSGD iteration along the lines of [19].

To observe the effect of coarse approximations of the optimal solution of (6), Fig. 2 shows the λ iterates (averaged over the last N_{train} iterations), vs. the number of PGD iterations, for different tolerance values. Although coarser approximations entailed fewer inner iterations, there was no significant impact on the overall convergence time.

Concluding remarks: In this paper, the hyper-subgradient of the LOO validation error has been derived for regression problems with non-smooth regularizers, exploiting the structure of PGD. The HSGD algorithm has been proposed, together with two approximations, namely a (stochastic) online variant, and an inexact update; and the numerical results corroborate the reduction in computation.

The flexibility of this approach to accomodate any convex, proximable regularizer enables extensions to generalizations of Lasso, which will be developed in future work.

REFERENCES

[1] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Advances Neural Inf. Process. Syst.*, 2011, pp. 2546–2554.

[2] D. Homrighausen and D. J. McDonald, "Leave-one-out cross-validation is risk consistent for lasso," *Machine learning*, vol. 97, no. 1-2, pp. 65–78, 2014.

[3] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning: Methods, Systems, Challenges*, Cham, 2019, pp. 3–33, Springer International Publishing.

[4] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012.

[5] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6765–6816, 2018.

[6] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyperparameters on large datasets," in *Artificial Intelligence and Stat.*, 2017, pp. 528–536.

[7] F. Pedregosa, "Hyperparameter optimization with approximate gradient," *arXiv preprint arXiv:1602.02355*, 2016.

[8] R. P. Monti, C. Anagnostopoulos, and G. Montana, "Adaptive regularization for lasso models in the context of nonstationary data streams," *Stat. Analysis and Data Mining: The ASA Data Science Journal*, vol. 11, no. 5, pp. 237–247, 2018.

[9] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, vol. 70, pp. 1165–1173.

[10] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1568–1577.

[11] J. Lorraine and D. Duvenaud, "Stochastic hyperparameter optimization through hypernetworks," *arXiv preprint arXiv:1802.09419*, 2018.

[12] D. Ito, S. Takabe, and T. Wadayama, "Trainable ista for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3113–3125, 2019.

[13] S. Wang, W. Zhou, A. Maleki, H. Lu, and V. Mirrokni, "Approximate leave-one-out for high-dimensional non-differentiable learning problems," *arXiv preprint arXiv:1810.02716*, 2018.

[14] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International Conference on Machine Learning*, 2015, pp. 2113–2122.

[15] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[16] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[17] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathem.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[18] A. T. Puig, A. Wiesel, G. Fleury, and A. O. Hero, "Multidimensional shrinkage-thresholding operator and group lasso penalties," *IEEE Signal Processing Letters*, vol. 18, no. 6, pp. 363–366, 2011.

[19] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.