DISTRIBUTED LASSO FOR IN-NETWORK LINEAR REGRESSION*

Juan Andrés Bazerque, Gonzalo Mateos and Georgios B. Giannakis

Dept. of ECE, Univ. of Minnesota, 200 Union Street SE, Minneapolis, MN 55455, USA

ABSTRACT

The least-absolute shrinkage and selection operator (Lasso) is a popular tool for joint estimation and continuous variable selection, especially well-suited for the under-determined but sparse linear regression problems. This paper develops an algorithm to estimate the regression coefficients via Lasso when the training data is distributed across different agents, and their communication to a central processing unit is prohibited for e.g., communication cost or privacy reasons. The novel distributed algorithm is obtained after reformulating the Lasso into a separable form, which is iteratively minimized using the alternating-direction method of multipliers so as to gain the desired degree of parallelization. The per agent estimate updates are given by simple soft-thresholding operations, and inter-agent communication overhead remains at affordable level. Without exchanging elements from the different training sets, the local estimates provably *consent* to the global Lasso solution, i.e., the fit that would be obtained if the entire data set were centrally available. Numerical experiments corroborate the convergence and global optimality of the proposed distributed scheme.

***Index Terms*—**Distributed estimation, Lasso, sparse regression.

1. INTRODUCTION

The least-absolute shrinkage and selection operator [1], best known as the *Lasso*, is a regularization technique capable of performing both estimation and continuous variable selection in linear regression problems. It combines the features of ridge regression and subset selection – standard techniques traditionally utilized to improve the least-squares (LS) estimates; see e.g. [2, p. 57]. The Lasso yields

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \qquad (1)$$

where the ℓ_1 (ℓ_2^2) norm of a vector is defined as the sum of absolute values (squares) of its entries; the training data set $\{y_l, \mathbf{x}_l\}_{l=1}^N$ is collected in the vector $\mathbf{y} := [y_1 \dots y_N]^T \in \mathbb{R}^N$; and the matrix $\mathbf{X} := [\mathbf{x}_1 \dots \mathbf{x}_N]^T \in \mathbb{R}^{N\times p}$. The nonzero coefficients in the estimate $\hat{\boldsymbol{\beta}}_{\text{lasso}}$ indicate which variables are relevant in the resulting linear predictor $y = \mathbf{x}^{\mathbf{T}}\hat{\boldsymbol{\beta}}_{\text{lasso}}$. Parameter $\lambda \geq 0$ controls the amount of shrinkage over $\hat{\boldsymbol{\beta}}_{\text{lasso}}$ effected by the ℓ_1-norm sparsity-encouraging penalty, and is typically chosen via model selection techniques such as cross-validation (CV); see e.g., [2]. Problem (1) is also known as *basis pursuit denoising* [3], in the context of finding the best sparse signal expansion using an overcomplete basis set.

*Work in this paper was supported by the NSF grants CCF-0830480 and ECCS-0824007; and also through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

Finding $\hat{\boldsymbol{\beta}}_{\text{lasso}}$ requires solving (iteratively) for any given value of λ a quadratic program (QP) with linear constraints. While standard QP solvers can be invoked to this end, an increasing amount of effort has been put recently into developing fast algorithms that capitalize on the unique properties of the Lasso; see, e.g. [4], [5], [6] and references therein.

Typically, the training set is assumed to be centrally available, so that it can be jointly processed to obtain $\hat{\boldsymbol{\beta}}_{\text{lasso}}$. However, collecting all data in a central location or fusion center (FC) may be prohibitive in various applications of interest. Distributed linear regression problems commonly arise with wireless sensor networks (WSNs), where data is inherently scattered across a large geographical area under surveillance. Sensors are battery operated, thus transferring all data to an FC (possibly located far away) may be infeasible due to power constraints imposed on the individual nodes. In-network-based Lasso estimators also find application in the sensing task of cognitive radio networks [7]. In other cases such as the Internet or collaborative inter-laboratory studies, agents providing private data for the purpose of e.g., fitting a sparse model, may not be willing to share their training data but only the learning results [8].

Having this context in mind, the present paper develops a consensus-based distributed algorithm for the Lasso. The novel approach entails reformulating (1) as a convex constrained optimization problem, whose structure lends itself naturally to distributed implementation. It is then possible to capitalize on this favorable structure by resorting to the alternating-direction method of multipliers (AD-MoM), an iterative optimization method especially well-suited for parallel processing [9, p. 253]. This way, provably convergent local recursions are derived to update each agent's local estimate, as well as a vector of dual prices responsible of effecting agreement across all agents. Thanks to the constrained reformulation of (1), the updates are obtained in closed form by simple soft-thresholding operations. On a per iteration basis, agents only exchange their current local estimate with their neighbors, so that the training data efficiently percolates to all agents without compromising its secrecy. Finally, a distributed CV procedure is developed to select the best λ in (1), in the sense of minimizing an estimate of the expected prediction error. The algorithm exploits "warm starts" to efficiently compute the Lasso path of solutions over a grid of values for λ [4].

2. PRELIMINARIES AND PROBLEM STATEMENT

Consider J networked agents that are capable of performing some local computations, as well as exchanging messages among neighbors. An agent should be understood as an abstract entity, possibly representing a sensor node in a WSN, a router monitoring Internet traffic, a hospital, insurance company or laboratory involved in e.g., a medical study; or a sensing radio from a next-generation mobile communications technology. The network is naturally modeled as an undirected graph $\mathcal{G}(\mathcal{J}, \mathcal{E})$, where the vertex set $\mathcal{J} := \{1, \dots, J\}$

corresponds to the agents, and the edges in $\mathcal{E}$ represent pairs of agents that can communicate. Agent $j \in \mathcal{J}$ communicates with its single-hop neighboring agents in $\mathcal{N}_j$, and the size of the neighborhood is denoted by $|\mathcal{N}_j|$.

The graph $\mathcal{G}$ is assumed connected, i.e., there exists a (possibly multihop) path that joins any pair of agents in the network.

Each agent $j \in \mathcal{J}$ has available a local training data set $\{y_l, \mathbf{x}_l\}_{l=1}^{N_j} = \{\mathbf{y}_j, \mathbf{X}_j\}$ of size N_j. Agents collaborate to form the estimator (1) in a distributed fashion, which can be rewritten as

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\sum_{j=1}^{J} \|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{2}$$

where $\mathbf{y} := [\mathbf{y}_1^T \ldots \mathbf{y}_J^T]^T \in \mathbb{R}^{N\times 1}$ with $N := \sum_{j=1}^{J} N_j$ and $\mathbf{X} := [\mathbf{X}_1^T \ldots \mathbf{X}_J^T]^T \in \mathbb{R}^{N\times p}$.

The objective of this paper is to develop and analyze in terms of convergence, a distributed algorithm for the Lasso based on in-network processing of the locally available training data. The described setup naturally suggests three characteristics that the algorithm should exhibit: (i) convergence to the global solution $\hat{\boldsymbol{\beta}}_{\text{lasso}}$ in (2); (ii) processing per agent should be kept as simple and efficient as possible; and (iii) communications among agents should be confined to the single-hop neighbors, and avoid exchanges of elements from the different training sets.

3. DISTRIBUTED ESTIMATION VIA LASSO

This section introduces the D-Lasso algorithm, going through the algorithmic construction steps, and salient features of its operation. Its convergence to the global estimator $\hat{\boldsymbol{\beta}}_{\text{lasso}}$ is also established.

3.1. A consensus-based reformulation of the Lasso

To distribute the cost in (2), replace the *global* variable $\boldsymbol{\beta}$ which couples the per-agent summands with *local* variables $\{\boldsymbol{\beta}_j\}_{j=1}^{J}$ representing candidate estimates of $\boldsymbol{\beta}$ per agent. It is now possible to reformulate (2) as a convex *constrained* minimization problem:

$$\left\{\hat{\boldsymbol{\beta}}_j\right\}_{j=1}^{J} = \arg\min_{\{\boldsymbol{\beta}_j\}} \frac{1}{2}\sum_{j=1}^{J}\left[\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right] \tag{3}$$
$$\text{s. t.} \quad \boldsymbol{\beta}_j = \boldsymbol{\beta}_i,\ j \in \mathcal{J},\ i \in \mathcal{N}_j.$$

The equality constraints directly effect local agreement across each agent's neighborhood. If the communication graph $\mathcal{G}$ is further assumed *connected*, these constraints also ensure *global* consensus a fortiori, meaning that $\boldsymbol{\beta}_j = \boldsymbol{\beta}_{j'}, \forall j, j' \in \mathcal{J}$. Indeed, let $P(j, j') : j, j_1, j_2, \ldots, j_n, j'$ be a path on $\mathcal{G}$ that joins an arbitrary pair of agents (j, j'). Because contiguous agents in the path are neighbors by definition, the corresponding chain of equalities $\boldsymbol{\beta}_j = \boldsymbol{\beta}_{j_1} = \boldsymbol{\beta}_{j_2} = \ldots = \boldsymbol{\beta}_{j_n} = \boldsymbol{\beta}_{j'}$ effected by the constraints in (3) imply $\boldsymbol{\beta}_j = \boldsymbol{\beta}_{j'}$ as desired. Thus, the constraints can be eliminated by replacing all the $\{\boldsymbol{\beta}_j\}$ with a common $\boldsymbol{\beta}$, in which case the cost in (3) reduces to the one in (2). The previous discussion has established the following important result.

Proposition 1: *If $\mathcal{G}$ is a connected graph,* (2) *and* (3) *are equivalent optimization problems, in the sense that $\hat{\boldsymbol{\beta}}_{lasso} = \hat{\boldsymbol{\beta}}_j,\ \forall\, j \in \mathcal{J}$.*

3.2. The D-Lasso algorithm

In order to tackle (3) in a distributed fashion, we will resort to the AD-MoM [9]. To this end, consider first the auxiliary local variables $\{\{\tilde{\boldsymbol{\gamma}}_j^i\}_{i\in\mathcal{N}_j}, \{\check{\boldsymbol{\gamma}}_j^i\}_{i\in\mathcal{N}_j}\}_{j\in\mathcal{J}}$, and replace the constraints in (3) with the equivalent ones $\boldsymbol{\beta}_j = \tilde{\boldsymbol{\gamma}}_j^i,\ \boldsymbol{\beta}_i = \check{\boldsymbol{\gamma}}_j^i,\ \tilde{\boldsymbol{\gamma}}_j^i = \check{\boldsymbol{\gamma}}_j^i,\ j \in \mathcal{J},\ i \in \mathcal{N}_j$. Variables $\{\tilde{\boldsymbol{\gamma}}_j^i, \check{\boldsymbol{\gamma}}_j^i\}$ are only used to derive the local recursions but will be eventually eliminated.

Next, consider the additional group of auxiliary local variables $\{\boldsymbol{\gamma}_j\}_{j=1}^{J}$, one per agent. Through them, the goal is to split the cost in (3) so that the squared error loss depends on the $\{\boldsymbol{\gamma}_j\}$, while the ℓ_1-norm penalty is a function of the variables $\{\boldsymbol{\beta}_j\}$. This way, the optimizations with respect to (w.r.t.) $\{\boldsymbol{\beta}_j\}$ will be shown to boil down to a Lasso in the so-termed orthonormal regression setup, which corresponds to having $\mathbf{X}^T\mathbf{X} = \mathbf{I}_p$ in (1), where $\mathbf{I}_p$ denotes the $p\times p$ identity matrix. Solutions in this case become available in closed form, in the form of soft-thresholding operations [2, p. 69]. A similar idea was applied to ℓ_1-norm regularized problems in [5]. While introducing appropriate constraints $\boldsymbol{\beta}_j = \boldsymbol{\gamma}_j$ that guarantee the equivalence of the formulations, (3) can be recast as

$$\min_{\{\boldsymbol{\beta}_j\},\boldsymbol{\gamma}} \frac{1}{2}\sum_{j=1}^{J}\left[\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\gamma}_j\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right] \tag{4}$$
$$\text{s. t.} \quad \boldsymbol{\beta}_j = \boldsymbol{\gamma}_j,\ j \in \mathcal{J}$$
$$\boldsymbol{\beta}_j = \tilde{\boldsymbol{\gamma}}_j^i,\ \boldsymbol{\beta}_i = \check{\boldsymbol{\gamma}}_j^i,\ \tilde{\boldsymbol{\gamma}}_j^i = \check{\boldsymbol{\gamma}}_j^i,\ j \in \mathcal{J},\ i \in \mathcal{N}_j$$

where $\boldsymbol{\gamma} := \{\boldsymbol{\gamma}_j, \{\tilde{\boldsymbol{\gamma}}_j^i\}_{i\in\mathcal{N}_j}, \{\check{\boldsymbol{\gamma}}_j^i\}_{i\in\mathcal{N}_j}\}_{j\in\mathcal{J}}$ for notational convenience. As a consequence of Proposition 1, the optimal solutions of (4) correspond to $\hat{\boldsymbol{\beta}}_{\text{lasso}}$ across all agents. Different from (2) however, (4) has a separable structure that facilitates distributed implementation. To capitalize on this favorable structure, associate Lagrange multipliers $\boldsymbol{u} := \{\mathbf{u}_j, \{\tilde{\mathbf{u}}_j^i\}_{i\in\mathcal{N}_j}, \{\check{\mathbf{u}}_j^i\}_{i\in\mathcal{N}_j}\}_{j\in\mathcal{J}}$ with the constraints in (4), and form the augmented Lagrangian function

$$\begin{aligned}\mathcal{L}_a[\{\boldsymbol{\beta}_j\},\boldsymbol{\gamma},\boldsymbol{u}] =& \frac{1}{2}\sum_{j=1}^{J}\left[\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\gamma}_j\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right]\\ &+ \sum_{j=1}^{J}\mathbf{u}_j^T(\boldsymbol{\beta}_j - \boldsymbol{\gamma}_j) + \frac{c}{2}\sum_{j=1}^{J}\|\boldsymbol{\beta}_j - \boldsymbol{\gamma}_j\|_2^2\\ &+ \sum_{j=1}^{J}\sum_{i\in\mathcal{N}_j}\left[(\tilde{\mathbf{u}}_j^i)^T(\boldsymbol{\beta}_j - \tilde{\boldsymbol{\gamma}}_j^i) + (\check{\mathbf{u}}_j^i)^T(\boldsymbol{\beta}_i - \check{\boldsymbol{\gamma}}_j^i)\right]\\ &+ \frac{c}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{N}_j}\left[\|\boldsymbol{\beta}_j - \tilde{\boldsymbol{\gamma}}_j^i\|_2^2 + \|\boldsymbol{\beta}_i - \check{\boldsymbol{\gamma}}_j^i\|_2^2\right].\end{aligned} \tag{5}$$

The constraints $\boldsymbol{\gamma} \in C_{\boldsymbol{\gamma}} := \{\boldsymbol{\gamma} : \tilde{\boldsymbol{\gamma}}_j^i = \check{\boldsymbol{\gamma}}_j^i,\ j \in \mathcal{J},\ i \in \mathcal{N}_j\}$ have not been dualized, while $c > 0$ is a preselected penalty coefficient.

The AD-MoM entails an iterative process comprising three steps per iteration $k = 0, 1, 2, \ldots$. First, the augmented Lagrangian is minimized w.r.t. the collection $\{\boldsymbol{\beta}_j\}$, considering the auxiliary variables in $\boldsymbol{\gamma}$ and multipliers in $\boldsymbol{u}$ as fixed parameters. The resulting minimizers define the updates $\{\boldsymbol{\beta}_j(k+1)\}$ corresponding to iteration $k+1$. A unique set of minimizers is guaranteed to exist, from the strict convexity of the augmented Lagrangian. The pertinent minimization problem decouples into J sub-problems

$$\begin{aligned}\boldsymbol{\beta}_j(k+1) = \arg\min_{\boldsymbol{\beta}_j}\Bigg\{&\frac{\lambda}{J}\|\boldsymbol{\beta}_j\|_1 + \mathbf{u}_j^T(k)\boldsymbol{\beta}_j + \frac{c}{2}\|\boldsymbol{\beta}_j - \boldsymbol{\gamma}_j(k)\|_2^2\\ &+ \sum_{i\in\mathcal{N}_j}\left[\tilde{\mathbf{u}}_j^i(k) + \check{\mathbf{u}}_i^j(k)\right]^T\boldsymbol{\beta}_j\\ &+ \frac{c}{2}\sum_{i\in\mathcal{N}_j}\left[\|\boldsymbol{\beta}_j - \tilde{\boldsymbol{\gamma}}_j^i(k)\|_2^2 + \|\boldsymbol{\beta}_j - \check{\boldsymbol{\gamma}}_i^j(k)\|_2^2\right]\Bigg\}\end{aligned} \tag{6}$$

Algorithm 1 : D-lasso

Initialize to zero $\{\boldsymbol{\beta}_j(0), \boldsymbol{\gamma}_j(0), \mathbf{p}_j(-1), \mathbf{u}_j(-1)\}_{j\in\mathcal{J}}$.
for $k = 0, 1, \ldots$ all $j \in \mathcal{J}$ **do**
  Transmit $\boldsymbol{\beta}_j(k)$ to neighbors in $\mathcal{N}_j$.
  Update $\mathbf{p}_j(k)$ via (7) and $\mathbf{u}_j(k)$ via (8).
  Update $\boldsymbol{\beta}_j(k+1)$ using (9).
  Update $\boldsymbol{\gamma}_j(k+1)$ using (10).
end for

that can be cast as an orthonormal Lasso (details in [10]). Due to the variable splitting procedure that led to (4) and the block-coordinate descent nature of the AD-MoM, the "undesirable" coupling term $\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\gamma}_j\|_2^2$ is not present in (6). For these reasons, it is possible to obtain $\boldsymbol{\beta}_j(k+1)$ in closed form as detailed next [cf. (9)].

Second, $\mathcal{L}_a$ is minimized w.r.t. $\boldsymbol{\gamma} \in C_\gamma$ while keeping all other variables fixed, to yield the updates $\boldsymbol{\gamma}(k+1)$. Finally, in the third step the Lagrange multipliers in $\boldsymbol{u}$ are updated via dual gradient ascent iterations [9], and the cycle of three steps is repeated for the $(k+2)nd$ iteration. The aforementioned procedure amounts to a block-coordinate descent method with dual variable updates. At each step while minimizing the augmented Lagrangian, the variables not being updated are treated as fixed parameters and substituted with their most up to date values.

The separability of the augmented Lagrangian in (5) comes in two flavors, first w.r.t. the variable groups $\{\boldsymbol{\beta}_j\}$ and $\boldsymbol{\gamma}$, as well as across agents $j \in \mathcal{J}$. This in turn leads to highly parallelized, simplified recursions corresponding to the aforementioned three steps. Specifically, the AD-MoM solver leads to the following recursions to be run locally at every agent (detailed derivations are in [10])

$$\mathbf{p}_j(k) = \mathbf{p}_j(k-1) + c\sum_{i\in\mathcal{N}_j}[\boldsymbol{\beta}_j(k) - \boldsymbol{\beta}_i(k)] \tag{7}$$

$$\mathbf{u}_j(k) = \mathbf{u}_j(k-1) + c[\boldsymbol{\beta}_j(k) - \boldsymbol{\gamma}_j(k)] \tag{8}$$

$$\boldsymbol{\beta}_j(k+1) = [c(2|\mathcal{N}_j|+1)]^{-1}\,\mathcal{S}\left(c\boldsymbol{\gamma}_j(k) - \mathbf{p}_j(k) - \mathbf{u}_j(k) + c\sum_{i\in\mathcal{N}_j}[\boldsymbol{\beta}_j(k) + \boldsymbol{\beta}_i(k)], \lambda/J\right) \tag{9}$$

$$\boldsymbol{\gamma}_j(k+1) = \left[c\mathbf{I}_p + \mathbf{X}_j^T\mathbf{X}_j\right]^{-1}\left(\mathbf{X}_j^T\mathbf{y}_j + c\boldsymbol{\beta}_j(k+1) + \mathbf{u}_j(k)\right) \tag{10}$$

where $\mathbf{p}_j(k) := 2\sum_{i\in\mathcal{N}_j}\tilde{\mathbf{u}}_j^i(k)$. The function $\mathcal{S} : \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}^p$ performs a coordinate-wise soft-thresholding operation, i.e., its ith coordinate is given by $[\mathcal{S}(\mathbf{z},\mu)]_i = \text{sign}([\mathbf{z}]_i)\,(|[\mathbf{z}]_i| - \mu)_+$ where $(\cdot)_+ := \max(0,\cdot)$. Recursions (7)-(10) comprise the novel D-Lasso algorithm, tabulated as Algorithm 1. As promised, the inherently redundant auxiliary variables $\{\tilde{\boldsymbol{\gamma}}_j^i, \breve{\boldsymbol{\gamma}}_j^i, \breve{\mathbf{u}}_j^i\}$ have been eliminated. Each agent, say the jth, does not need to *separately* keep track of all its multipliers $\{\tilde{\mathbf{u}}_j^i(k)\}_{i\in\mathcal{N}_j}$, but only to update their sum $\mathbf{p}_j(k)$.

The algorithm entails the following steps. During iteration $k+1$, agent j receives the local estimates $\{\boldsymbol{\beta}_i(k)\}_{i\in\mathcal{N}_j}$ from its neighbors and plugs them into (7) to evaluate the dual price vector $\mathbf{p}_j(k)$. The new multiplier $\mathbf{u}_j(k)$ is then obtained using the locally available vectors $\{\boldsymbol{\gamma}_j(k), \boldsymbol{\beta}_j(k)\}$. Subsequently, vectors $\{\mathbf{p}_j(k), \mathbf{u}_j(k)\}$ are jointly used along with $\{\boldsymbol{\beta}_i(k)\}_{i\in\mathcal{N}_j}$ to obtain $\boldsymbol{\beta}_j(k+1)$ via the soft-thresholding/proportional shrinkage operation in (9). Finally, the updated $\boldsymbol{\gamma}_j(k+1)$ is obtained from (10) using the locally available training data. The $(k+1)$st iteration is concluded after agent j broadcasts $\boldsymbol{\beta}_j(k+1)$ to its neighbors.

Agents only exchange their sparse local estimates with their neighbors, and the communication cost does not depend on the size of the local training sets. The inversion of the $p \times p$ matrix $c\mathbf{I}_p + \mathbf{X}_j^T\mathbf{X}_j$ in (10) as well as the evaluation of the local "ridge" estimate $[c\mathbf{I}_p + \mathbf{X}_j^T\mathbf{X}_j]^{-1}\mathbf{X}_j^T\mathbf{y}_j$ have to be performed only once, and can be done offline before D-Lasso is run. Given these quantities, the updates (7)-(10) solely involve scaling/addition of sparse p-dimensional vectors and a soft-thresholding operation in (9).

As asserted in the ensuing proposition, D-Lasso generates local iterates $\boldsymbol{\beta}_j(k)$ that converge to the global Lasso. The proof in [10] amounts to checking that the qualification conditions for the convergence of the AD-MoM (as per Proposition 4.2 in [9, p. 257]) are satisfied by the optimization problem (4).

Proposition 2: *Let $\mathcal{G}$ be a connected graph and consider the D-Lasso recursions* (7)-(10)*. Then, for any value of the penalty coefficient $c > 0$, the iterates $\boldsymbol{\beta}_j(k)$ converge to the Lasso [cf.* (2)*], i.e.,* $\lim_{k\to\infty}\boldsymbol{\beta}_j(k) = \hat{\boldsymbol{\beta}}_{lasso}$, $\forall\, j \in \mathcal{J}$.

If the number of parameters p is greater than the size of the data set N, then a unique solution of (2) is not guaranteed for a general $\mathbf{X}$. Proposition 2 remains valid however, if the equality is replaced by a set of minimizers; that is,

$$\lim_{k\to\infty}\boldsymbol{\beta}_j(k) \in \arg\min_{\boldsymbol{\beta}}\frac{1}{2}\sum_{j=1}^{J}\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1.$$

4. DISTRIBUTED CROSS-VALIDATION

The D-Lasso algorithm in Section 3.2 assumes knowledge of the regularization parameter λ. This section introduces an algorithm to select λ, by performing CV in a distributed fashion. The training data is not flooded across the network, and only local exchanges (within the neighborhood) are required to this end.

The goal is to select the best parameter λ from a grid of candidate values $\lambda \in \Lambda := \{\lambda_1, \ldots, \lambda_L\}$, ordered as $\lambda_1 > \lambda_2 > \ldots > \lambda_L$, by properly modifying the leave-one-out CV technique (see e.g., [2].) For this purpose, each agent $j \in \mathcal{J}$ is set aside at a time and the rest of the agents $i \in \mathcal{J}^{(-j)} := \{i \in \mathcal{J}, i \neq j\}$ collaborate to obtain $\hat{\boldsymbol{\beta}}_i^{(-j)}$ by running the D-Lasso algorithm. After convergence, the common $\hat{\boldsymbol{\beta}}^{(-j)} := \hat{\boldsymbol{\beta}}_i^{(-j)}$ is communicated to agent j who forms the prediction error estimate based on its set aside data, i.e., $e_j(\lambda) := ||\mathbf{y}_j - \mathbf{X}_j\hat{\boldsymbol{\beta}}^{(-j)}||_2^2$. The same process is repeated for all $\lambda = \lambda_1, \ldots, \lambda_L$ using "warm starts" as in [4], and the results are stored in the $L \times 1$ vector $\mathbf{e}_j := [e_j(\lambda_1)\ldots e_j(\lambda_L)]^T$. Simulations suggest that with this type of initialization, convergence can be attained after a few iterations provided consecutive values of λ are not too far apart.

After repeating this procedure for all $\lambda \in \Lambda$ and all $j \in \mathcal{J}$, each agent has available the error vector $\mathbf{e}_j$. The decision on which λ to use is based on the average of these errors across nodes $j \in \mathcal{J}$.

$$\mathbf{e} := \frac{1}{J}\sum_{j=1}^{J}\mathbf{e}_j.$$

This average can be carried out using consensus averaging algorithms; see e.g., [11]. Once the vector $\mathbf{e} := [e(\lambda_1)\ldots e(\lambda_L)]^T$ becomes available to all agents, they can independently select the best λ as the one corresponding to the minimum entry of $\mathbf{e}$, i.e,

$$\lambda_{\text{dcv}} := \arg\min_{\lambda_l\in\Lambda}\{e(\lambda_l)\}_{l=1}^{L}.$$

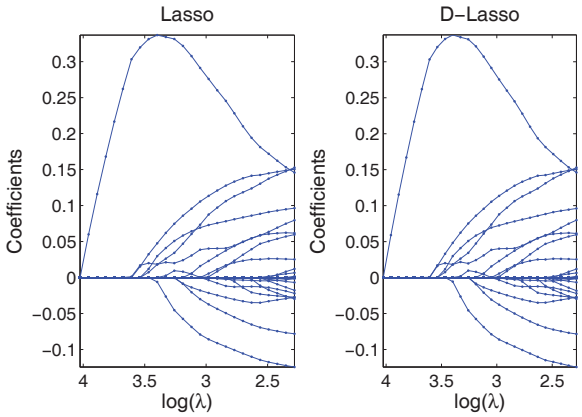

Fig. 1. SRBCT data: profiles of estimated coefficients for the centralized Lasso (left) and using the D-Lasso algorithm (right). For the latter, the path of solutions of a representative agent is shown.

Note that $\lambda_{\rm dcv}$ coincides with its centralized counterpart, that would be obtained by a standard CV scheme when applied to the entire data set using the same partition of the data in J folds. Generalized variants of this distributed CV technique are also possible by clustering agents in groups that are set aside one at a time, possibly removing part of the data of each user [10].

5. NUMERICAL EXAMPLE

A simulated test is now presented to corroborate the convergence of the D-Lasso algorithm. The data is taken from the example in [2, p. 651], which consists of $N = 63$ samples of $p = 2308$ genes measured with microarrays. The entries of $\mathbf{y}$ take values in $\{1, 2, 3, 4\}$, corresponding to classes of small round blue-cell tumors (SRBCT) found in children. For the purpose of illustration, the response variable is treated here as quantitative so that this becomes a highly underdetermined linear regression problem. The network of $J = 10$ agents is simulated as a random geometric graph on the unity square, with communication range $r = 0.3$. Data is randomly split across agents, so that $N_j = 6$ for $j \in [1, 9]$ and $N_{10} = 9$.

By running the D-Lasso algorithm and using "warm starts", the path of Lasso solutions is computed at 100 different values of the regularization parameter λ. The values are evenly spaced on a logarithmic scale in the interval $[10^{-4}\lambda_{\max}, \lambda_{\max}]$, where $\lambda_{\max}$ is the minimum value such that $\hat{\boldsymbol{\beta}}_{\rm lasso} \neq \mathbf{0}$ [4]. The penalty coefficient is set to $c = 7$, since several experiments suggested this value leads to fastest convergence. For agent $j = 3$, the results for the first 25 values of $\log(\lambda)$ are depicted on the right plot in Fig. 1. To corroborate that consensus is achieved to the desired global estimator, the path of solutions is also computed (centrally) via standard second-order cone programming software. The resulting regularization paths are shown on the left plot in Fig. 1, and are essentially identical to the ones obtained using the D-Lasso algorithm. As λ decreases, so does the amount of shrinkage, and more variables enter the model. The piecewise-linear coefficient profile appears as piecewise-nonlinear because $\log(\lambda)$ is used in the abscissa.

The leave-one-agent-out CV (in this case also ten-fold CV) procedure in Section 4 yields an optimum value of $\lambda_{\rm dcv} = 0.4874$, where only 32 variables enter the model. The Lasso selects those genes with the strongest effects, a feature particularly attractive when it comes to interpretation of high-dimensional data.

6. CONCLUDING REMARKS

An in-network processing-based algorithm for fitting a Lasso model is developed in this paper, that is suitable for distributed, sparse, linear regression tasks. The Lasso is reformulated into an equivalent constrained form, whose structure lends itself naturally to distributed implementation via the alternating-direction method of multipliers. Apart from an offline matrix inversion, the resulting per agent D-Lasso updates are extremely simple: just linear combinations of vectors and soft-thresholding operations. Agents only exchange sparse messages within the neighborhood, and the communication cost is independent of the size of the local training sets. The sequences of local estimates generated by the D-Lasso algorithm are provably convergent to $\hat{\boldsymbol{\beta}}_{\rm lasso}$, while a distributed cross-validation procedure is proposed to select the tuning parameter λ.

The framework and techniques introduced here to develop the D-Lasso algorithm are readily applicable to other related models. These include the adaptive Lasso that guarantees consistency of estimation and variable selection, the elastic net for correlated variables, and the smoothness-encouraging fused Lasso; see e.g. [2][1].

7. REFERENCES

[1] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal. Statist. Soc. B*, vol. 58, pp. 267–288, 1996.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, second edition, 2009.

[3] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.

[4] J. Friedman, T. Hastie, H. Hofling, and R. Tibshirani, "Pathwise coordinate optimization," *Ann. Appl. Stat.*, vol. 1, pp. 302–332, 2007.

[5] T. Goldstein and S. Osher, "The split bregman method for l1 regularized problems," *UCLA CAM Report 08-29*.

[6] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Trans. Signal Process.*, vol. 57, pp. 2479–2493, 2009.

[7] J.-A Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Process.*, 2010 (to appear).

[8] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations*, vol. 4, 2003.

[9] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena-Scientific, second edition, 1999.

[10] G. Mateos, J.-A Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *Ann. Appl. Stat.*, (submitted).

[11] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[1]The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies of the Army Research Laboratory or the U. S. Government.