SPARSITY-PROMOTING ADAPTIVE ALGORITHM FOR DISTRIBUTED LEARNING IN DIFFUSION NETWORKS

Symeon Chouvardas[1] *Konstantinos Slavakis*[2] *Yannis Kopsinis*[1] *Sergios Theodoridis*[1]

[1]University of Athens,
Dept. of Informatics and Telecommunications,
Athens 15784, Greece.
Emails: schouv@di.uoa.gr, kopsinis@ieee.org, stheodor@di.uoa.gr

[2]University of Peloponnese,
Dept. of Telecommunications Science and Technology,
Tripolis 22100, Greece.
Email: slavakis@uop.gr

ABSTRACT

In this paper, a sparsity-promoting adaptive algorithm for distributed learning in diffusion networks is developed. The algorithm follows the set-theoretic estimation rationale, i.e., at each time instant and at each node, a closed convex set, namely a hyperslab, is constructed around the current measurement point. This defines the region in which the solution lies. The algorithm seeks a solution in the intersection of these hyperslabs by a sequence of projections. Sparsity is encouraged in two complimentary ways: a) by employing extra projections onto a weighted ℓ_1 ball, that complies with our desire to constrain the respective weighted ℓ_1 norm and b) by adopting variable metric projections onto the hyperslabs, which implicitly quantify data mismatch. A combine-adapt cooperation strategy is adopted. Under some mild assumptions, the scheme enjoys a number of elegant convergence properties. Finally, numerical examples verify the validity of the proposed scheme, compared to other algorithms, which have been developed in the context of sparse adaptive learning.

Index Terms— Adaptive distributed learning, sparsity, diffusion networks, projections.

1. INTRODUCTION

Sparse signal estimation has been recently attracting an overwhelming interest, mainly, under the compressive sensing framework. The vast majority of such algorithms are batch solvers, which implies that an estimate is obtained, once a fixed number of measurements is collected and stored. However, scenarios where the data are sequentially received and/or the unknown parameter vector is time-varying, cannot be treated by batch algorithms, since this would imply excessive processing and memory requirements. Online, sparsity-promoting adaptive learning techniques overcome such limitations. Algorithms that belong to this category update the estimate at each time instance, exploiting the newly received measurements. Moreover, the a-priori information, concerning the sparsity of the unknown parameter vector, is embedded by employing sparsity promoting constraints, which usually revolve around the ℓ_1 norm of the unknown parameter vector, e.g., [1, 2].

In this paper, we study the problem of adaptive distributed learning [3, 4]. Although, there are a few sparsity-promoting algorithms for distributed learning for the batch scenario setting, e.g., [5], to our knowledge, this is the first time that an adaptive, sparsity-promoting algorithm for distributed learning is developed. A network of sensors is considered and the task is to estimate a parameter vector, which is *common* to all nodes, based on the noisy measurements that are received *locally* at each one of the nodes. A first approach would be the so-called centralized solution. In such a scenario, the nodes transmit the received information, to a central node, called fusion center, which then takes over to carry out the full amount of computations. This scheme is not always feasible to be adopted, due to power and/or geographical constraints. Moreover, it lacks robustness, since if the fusion center fails, the whole network collapses. For these reasons, in many applications, a fully decentralized methodology is followed, and each node performs computations locally, according to a predefined protocol. There are mainly two topologies; a) the incremental, in which each node communicates with only one neighbouring node and the network results to a ring topology and b) the diffusion, e.g., [4, 6, 7], where each node communicates with a number of neighbouring nodes, which define its neighbourhood. In this paper, the diffusion topology is considered, since it is more robust to node failures, compared to the incremental one; in the latter topology, if a node is malfunctioning the network collapses. Moreover, the implementation of the diffusion topology turns out to be easier when large networks are involved.

Our novel algorithm is developed within the set-theoretic estimation rationale and, more specifically, it is based on projections onto convex sets. At each time instance, a closed convex set, namely a hyperslab, is constructed, based on the received measurements, and one seeks for a solution within this set. Moreover, in order to impose sparsity on the unknown vector, projections onto sparsity-promoting weighted ℓ_1 balls, [1], take place. Our desire for sparsity is further strengthened, by reformulating the projection operators appropriately. To this end (see, for example, [8]), we adopt the variable metric projections rationale. In principle, the variable metric projections improve the convergence speed, when seeking for a sparse vector, due to the fact that different weights are assigned at each coefficient of the updated vector, and, through this procedure, small coefficients are pushed to diminish faster. The rationale of assigning different weights at each coefficient is also met in the so-called proportionate algorithms, [9, 10]. The proposed algorithmic scheme enjoys a number of nice convergence properties, such as, monotonicity, asymptotic optimality and strong convergence to a point that satisfies the consensus property, and the performance of the proposed algorithm is validated, via a system identification task.

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

2. SPARSITY-AWARE ADAPTIVE LEARNING

The set of all real numbers will be denoted by $\mathbb{R}$. The stage of discussion will be the Euclidean space $\mathbb{R}^m$, where m is a positive integer. Vectors and matrices will be denoted by boldface, and uppercase boldface letters respectively, and the symbol $(\cdot)^T$ will stand for the transpose of a vector. Given a positive definite matrix, say $\boldsymbol{V}$, of dimension $m \times m$, we define the weighted inner product as follows: $\forall \boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathbb{R}^m$, $\langle \boldsymbol{h}_1, \boldsymbol{h}_2 \rangle_{\boldsymbol{V}} = \boldsymbol{h}_1^T \boldsymbol{V} \boldsymbol{h}_2$ and the weighted norm, $\forall \boldsymbol{h} \in \mathbb{R}^m$, $\|\boldsymbol{h}\|_{\boldsymbol{V}} = \sqrt{\boldsymbol{h}^T \boldsymbol{V} \boldsymbol{h}}$. The Euclidean norm, denoted by $\|\cdot\|$, is a special case of the weighted norm, and occurs if we let $\boldsymbol{V} = \boldsymbol{I}_m$, where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix. Finally, given a vector $\boldsymbol{h} = [h_1, \ldots, h_m]^T \in \mathbb{R}^m$, the ℓ_1 norm is defined as $\|\boldsymbol{h}\|_1 = \sum_{i=1}^m |h_i|$, and the support set, $\operatorname{supp}(\boldsymbol{h}) := \{i \in 1, \ldots, m : h_i \neq 0\}$. The ℓ_0 (pseudo) norm, i.e., $\|\boldsymbol{h}\|_0$, denotes the number of non-zero coefficients of $\boldsymbol{h}$.

We consider the problem of estimating an unknown parameter vector $\boldsymbol{h}_*$, through measurements $(d_n, \boldsymbol{u}_n) \in \mathbb{R} \times \mathbb{R}^m$, which are related according to the linear system

$$d_n = \boldsymbol{u}_n^T \boldsymbol{h}_* + v_n, \tag{1}$$

where v_n is the noise process, with standard deviation equal to σ. The unknown vector is assumed to be sparse, i.e., $\|\boldsymbol{h}_*\|_0 \ll m$, or, in other words, it has a few number of non-zero coefficients[1]. Sparsity promoting, adaptive algorithms have been proposed in the literature, e.g., [1, 2]. In a nutshell, in such techniques, the effort is twofold. First, the estimate of the unknown parameter vector is obtained so that to minimize the misfit based on the output-input training data. At the same time, the learning process is assisted, via a constraint built around the ℓ_1 norm. It has been verified that by embedding this sparsity-promoting constraint, the algorithms converge significantly faster, and they rest at a lower steady state error floor, compared to the ones which obtain estimates by relying solely on the received measurement pairs.

2.1. Set theoretic estimation approach and variable metric projections

2.1.1. The set theoretic estimation approach

In the current study, the set theoretic estimation approach is followed. That is, instead of seeking for a vector that optimizes a certain loss function, we seek for points that are *in agreement* with the available measurements. More specifically, at each time instance, a closed convex set, namely hyperslab, defined as $S_n := \{\boldsymbol{h} \in \mathbb{R}^m : |d_n - \boldsymbol{u}_n^T \boldsymbol{h}| \leq \epsilon\}$, is constructed based on the current measurements d_n and $\boldsymbol{u}_n$; any point that lies in this set is considered to be in agreement with the current measurements. The user-defined parameter ϵ is chosen so as to take into consideration the noise. Parameters, such as ϵ, which determine the width of a set onto which one seeks for a candidate solution, are also met in the so-called set-membership algorithms [11]. So, the goal is to find a point that lies in the intersection of all the hyperslabs, which are built sequentially, one per time instance, as the data are collected. In order to achieve this, projections onto the hyperslabs, under a certain rule, take place, sequentially, in order to lead the produced estimates towards the required intersection, e.g., [1].

[1] The derivation can easily be extended for the cases where the unknown vector is not itself sparse but it accepts a sparse representation in some domain/dictionary.

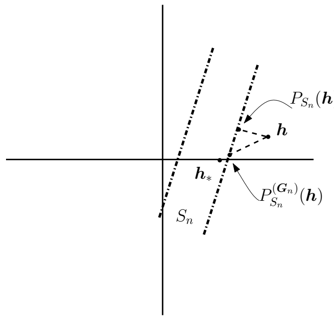

$P_{S_n}(\boldsymbol{h})$
$\boldsymbol{h}$
$\boldsymbol{h}_*$
$P_{S_n}^{(\boldsymbol{G}_n)}(\boldsymbol{h})$
S_n

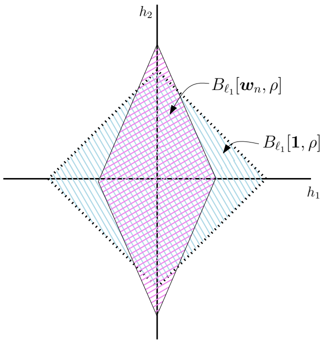
Fig. 1. Illustration of a hyperslab, the Euclidean projection of a vector $\boldsymbol{h}$ onto it, denoted by $P_{S_n}(\boldsymbol{h})$, and the variable metric projection onto it.

2.1.2. Employing sparsity promoting variable metric projections

The first step, at which our a-priori knowledge, about the underlying sparsity, is embedded into our algorithmic process, takes place via a specific choice of the projection operator; more specifically, the notion of the variable metric projections is adopted. Note that this step can be bypassed and leave the sparsity promotion only to "hands" of the ℓ_1 norm. However, the combination of the two leads to the best performance. The variable metric projection of an arbitrary point, with respect to a positive definite matrix $\boldsymbol{G}_n$, onto a hyperslab, $\forall \boldsymbol{h} \in \mathbb{R}^m$, is given as

$$P_{S_n}^{(\boldsymbol{G}_n)}(\boldsymbol{h}) = \boldsymbol{h} + \begin{cases} \dfrac{d_n - \boldsymbol{u}_n^T \boldsymbol{h} + \epsilon}{\|\boldsymbol{u}_n\|^2_{\boldsymbol{G}_n^{-1}}} \boldsymbol{G}_n^{-1} \boldsymbol{u}_n, & \text{if } d_n - \boldsymbol{u}_n^T \boldsymbol{h} < -\epsilon, \\ 0, & \text{if } |d_n - \boldsymbol{u}_n^T \boldsymbol{h}| \leq \epsilon, \\ \dfrac{d_n - \boldsymbol{u}_n^T \boldsymbol{h} - \epsilon}{\|\boldsymbol{u}_n\|^2_{\boldsymbol{G}_n^{-1}}} \boldsymbol{G}_n^{-1} \boldsymbol{u}_n, & \text{if } d_n - \boldsymbol{u}_n^T \boldsymbol{h} > \epsilon. \end{cases} \tag{2}$$

The standard Euclidean projection, onto a hyperslab, occurs if in the previous equation, we let $\boldsymbol{G}_n = \boldsymbol{I}_m$. Let us now shed some more light on the physical reasoning behind the variable metric projections. The positive definite diagonal matrix $\boldsymbol{G}_n^{-1}$ is constructed by following a similar rationale as in [8, 9]. In words, the i-th coefficient of its diagonal equals to $g_{i,n}^{-1} = \frac{1-\alpha}{m} + \alpha \frac{|h_i^{(n)}|}{\|\boldsymbol{h}_n\|_1}$, where $\alpha \in [0, 1)$ is a parameter, which determines the extend to which the sparsity level of the unknown vector will be taken into consideration, and $h_i^{(n)}$ denotes the i-th component of $\boldsymbol{h}_n$, which is the estimate at time instance n. To learn by example, consider the ideal situation, in which $\boldsymbol{G}_n^{-1}$ is constructed using the unknown vector $\boldsymbol{h}_*$. In that case, it can be seen that $g_{i,n}^{-1} > g_{i',n}^{-1}$, if $i \in \operatorname{supp}(\boldsymbol{h}_*)$, and $i' \notin \operatorname{supp}(\boldsymbol{h}_*)$. Moreover, notice that the amplitude of each coefficient of the vector, used to construct $\boldsymbol{G}_n^{-1}$, determines the weight that will be assigned to the corresponding coefficient of the second term of the right hand side in (2). Hence, from the previous, it can be readily seen that components with smaller amplitude are multiplied with small coefficients of $\boldsymbol{G}_n^{-1}$. Through this procedure, these coefficients diminish faster, and the convergence speed is accelerated, when seeking for a sparse vector, compared to the case where Euclidean projections are performed. Obviously, since $\boldsymbol{h}_*$ is unknown, in order to assign the previously mentioned weights, we rely on the available estimate of it, i.e., $\boldsymbol{h}_n$, at each time instance. Schematically, these concepts are illustrated in Fig. 1.



h_2
$B_{\ell_1}[\boldsymbol{w}_n, \rho]$
$B_{\ell_1}[\mathbf{1}, \rho]$
h_1

Fig. 2. Illustration of a weighted ℓ_1 ball and an unweighted ℓ_1 ball.

2.2. Sparsity-promoting adaptive algorithm based on projections onto weighted ℓ_1 balls

This section refers to the second step, in which sparsity is enforced. In [1], a sparsity promoting adaptive algorithm, based on set theoretic estimation arguments, was proposed. Besides the Euclidean projections on the hypeslabs, an extra projection was performed, at each time iteration, onto a weighted ℓ_1 ball in order to enforce sparsity. As a matter of fact, this extra projection is equivalent to a soft-thresholding operation. Given a vector of weights $\boldsymbol{w}_n = [w_1^{(n)}, \ldots, w_m^{(n)}]^T$, where $w_i^{(n)} > 0, \forall i = 1, \ldots, m$, and a positive radius, ρ, the weighted ℓ_1 ball is defined: $B_{\ell_1}[\boldsymbol{w}_n, \rho] := \{\boldsymbol{h} \in \mathbb{R}^m : \sum_{i=1}^m w_i^{(n)} |h_i| \leq \rho\}$. The classical ℓ_1 ball, occurs if $\boldsymbol{w}_n = \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^m$ is the vector of ones. The projection onto $B_{\ell_1}[\boldsymbol{w}_n, \rho]$, is performed in a finite number of steps, and it is given in [1, Theorem 1]. A possible strategy, in order to construct the vector of weights, as was suggested in [1], is the following: $w_i^{(n)} = 1/(|h_i^{(n)}| + \tilde{\epsilon}_n),\ i = 1, \ldots, m$, where $\tilde{\epsilon}_n$ is a sequence of positive numbers, used in order to avoid divisions by zero. Furthermore, it has been shown that a necessary condition in order to have $\boldsymbol{h}_* \in B_{\ell_1}[\boldsymbol{w}_n, \rho]$, is to choose the radius according to the following rule: $\rho \geq \|\boldsymbol{h}_*\|_0$. The geometry of the weighted and the classical ℓ_1 ball is illustrated in Fig. 2

Towards developing our new scheme, the algorithm in [1] was generalized to involve *variable metric* projections on the weighted ℓ_1 ball, too. This was necessary in order to have a common projection operator, both for the hypeslabs as well the weighted ℓ_1 ball. Besides faster convergence, such a common framework was necessary for the theoretical analysis of the resulting algorithm, in its distributed mode of operation.

Claim 1: The variable metric projection, with respect to the matrix $\boldsymbol{G}_n$, onto $B_{\ell_1}[\boldsymbol{w}_n, \rho]$ is given by $P^{(\boldsymbol{G}_n)}_{B_{\ell_1}[\boldsymbol{w}_n, \rho]} = \boldsymbol{G}_n^{-\frac{1}{2}} P_{B_{\ell_1}[\boldsymbol{G}_n^{-\frac{1}{2}} \boldsymbol{w}_n, \rho]} \boldsymbol{G}_n^{\frac{1}{2}}$.

Proof: The proof is omitted due to lack of space, and it will be presented elsewhere.□

3. ADAPTIVE DISTRIBUTED LEARNING

We now turn our focus onto the main part of our paper; this part blends the two previous steps in order to be used in a distributed processing context. Our task is to estimate a sparse, unknown parameter vector $\boldsymbol{h}_*$, exploiting measurements collected at the N nodes of a network in accordance to the diffusion topology, e.g., [3]. We denote the node set by $\mathcal{N} = \{1, \ldots, N\}$, and we assume that each node is able to exchange information, with a subset of $\mathcal{N}$, namely $\mathcal{N}_k \subseteq \mathcal{N}, k = 1, \ldots, N$. This set, is also known as the *neighbourhood* of k. Each node has access to the measurements $(d_{k,n}, \boldsymbol{u}_{k,n})$, which obey the linear model (1), and the standard deviation of the noise at each node, equals to σ_k. In adaptive distributed learning, a node, say k, in order to provide an estimate, exploits the information sensed by the environment, i.e., $d_{k,n}$ and $\boldsymbol{u}_{k,n}$, *as well as* the information received by the neighbourhood, that is, the estimates $\forall l \in \mathcal{N}_k$. It has been shown, e.g., [6], that if the nodes of the network cooperate, the performance of the respective algorithms is enhanced, compared to the case where each node operates individually. Moreover, this exchange of information can lead to asymptotic consensus, e.g., [4]; that is, the nodes will converge to the *same* estimate.

In this paper, we follow the combine-adapt cooperation strategy, in which, at every node, the estimates from the neighbourhood are fused under a certain protocol, and then the aggregate is put into the adaptation step, e.g., [3, 4]. To be more specific, node k assigns a positive weight at the estimates of each node of its neighbourhood, i.e., $\boldsymbol{h}_{l,n}, \forall l \in \mathcal{N}_k$, in order to compute the aggregate $\boldsymbol{\phi}_{k,n} := \sum_{l \in \mathcal{N}_k} c_{k,l} \boldsymbol{h}_{l,n}$, where the positive weights $c_{k,l}, \forall k \in \mathcal{N}, \forall l \in \mathcal{N}_k$ are called combination coefficients, and it holds that $c_{k,l} > 0, l \in \mathcal{N}_k$ and $\sum_{l \in \mathcal{N}_k} c_{k,l} = 1, \forall k \in \mathcal{N}$. After the combination step, i.e., the computation of $\boldsymbol{\phi}_{k,n}$, the latter term gets involved in the adaptation process.

4. THE PROPOSED ALGORITHM

Our goal is to bring together the methodologies, which were presented in section 2, and to reformulate them in a distributed fashion, by adopting the combine-adapt protocol. The main steps of the algorithm, can be summarized as follows:

1. At each node $k \in \mathcal{N}$, the estimates from the neighbourhood are received and fused in order to compute $\boldsymbol{\phi}_{k,n}$.
2. Exploiting the newly received measurements, $d_{k,n}, \boldsymbol{u}_{k,n}$ the following hyperslab is defined: $S_{k,n} = \{\boldsymbol{h} \in \mathbb{R}^m : |d_{k,n} - \boldsymbol{u}_{k,n}^T \boldsymbol{h}| \leq \epsilon_k\}$. The parameter ϵ_k is allowed to vary from node to node, according to the specificities of the respective noise source. The aggregate $\boldsymbol{\phi}_{k,n}$ is projected, using variable metric projections, onto the q most recent hyperslabs, constructed locally, and a convex combination of these q projections is computed. The effect of projecting onto a $q > 1$ number of hyperslabs is to speed up convergence, [1].
3. The result of the previous step is projected, via a metric projection, onto the sparsity promoting constraint set, i.e., the weighted ℓ_1 ball.

The previous steps can be written compactly in the following formula

$$\boldsymbol{h}_{k,n+1} = P^{(\boldsymbol{G}_n)}_{B_{\ell_1}[\boldsymbol{w}_n, \rho]} \left(\boldsymbol{\phi}_{k,n} + \mu_{k,n} \left(\sum_{j=n-q+1}^{n} \frac{1}{q} P^{(\boldsymbol{G}_n)}_{S_{k,j}}(\boldsymbol{\phi}_{k,n}) - \boldsymbol{\phi}_{k,n} \right) \right), \tag{3}$$

where $\mu_{k,n} \in (0, 2\mathcal{M}_{k,n})$, with

$$\mathcal{M}_{k,n} := \begin{cases} \dfrac{\sum_{j=n-q+1}^{n} \frac{1}{q} \|P^{(\boldsymbol{G}_n)}_{S_{k,j}}(\boldsymbol{\phi}_{k,n}) - \boldsymbol{\phi}_{k,n}\|^2}{\|\sum_{j=n-q+1}^{n} \frac{1}{q} P^{(\boldsymbol{G}_n)}_{S_j}(\boldsymbol{\phi}_{k,n}) - \boldsymbol{\phi}_{k,n}\|^2}, \\ \quad \text{if } \sum_{j=n-q+1}^{n} \frac{1}{q} P^{(\boldsymbol{G}_n)}_{S_{k,j}}(\boldsymbol{\phi}_{k,n}) \neq \boldsymbol{\phi}_{k,n} \\ 1, \quad \text{otherwise.} \end{cases}$$

In this work, we have shown that under some mild assumptions, the algorithm enjoys a number of nice properties, as for example: monotonicity, asymptotic optimality, which implies that asymptotically the distance of the estimates from the hyperslabs will tend to zero, asymptotic consensus, and strong convergence. It should be stressed out that since the algorithm follows the set-theoretic estimation rationale, the convergence proof is non trivial and it is built upon deterministic arguments, as opposed to diffusion algorithms, which are LMS-based, where the arguments are stochastic. Due to lack of space, the proof is omitted and will be presented elsewhere.

Remark 1: Notice from (3), that the weighted ℓ_1 ball and the matrix $\boldsymbol{G}_n$ are assumed to be common in all the nodes of the network. This is essential and it is required by the convergence proof, in order to guarantee asymptotic consensus. In practice, a reasonable strategy to achieve this would be to construct the weights $\boldsymbol{w}_n$ and $\boldsymbol{G}_n$, via $\boldsymbol{h}_{k_{opt},n}$ where k_{opt} is the node with the smallest noise variance. This requires knowledge, in every node, of $\boldsymbol{h}_{k_{opt},n}$, something that, in general, is infeasible. However, it turns out this not to be essential to update the parameters every time instance; instead, $\boldsymbol{w}_n$ and $\boldsymbol{G}_n$ can be updated at every n' time instances, where n' are the time steps required for $\boldsymbol{h}_{k_{opt},n}$ to be distributed over the network. However, as it will become clear in the numerical examples section, it turns out that the algorithm is robust in cases where the knowledge of the less noisy node is not available, and/or in cases where the assumption that these quantities must be common to all nodes is violated and each node uses the locally available values. It should be pointed out that such discrepancies in the adaptive filtering are common. For example, in the LMS, the independence assumption is commonly employed to prove convergence, although in practice this can be violated.

Remark 2: The complexity of the proposed algorithm, is of order $O(qm)$, coming from the projection onto the hyperslabs, and $O(m\log m)$, coming from the projection onto the weighted ℓ_1 ball [1].

Remark 3: The algorithm needs the following main user-defined parameters: a) ϵ; this is usually set equal to $\epsilon_k = \sqrt{2}\times\sigma_k$, although the algorithm is not very sensitive to it. b) q; the larger the q the faster the convergence. This corresponds to the number of subspaces used in the Affine Projection Algorithm (APA). This is not a critical parameter, and one can choose it depending on the complexity load that can be afforded by the algorithm in real time operations. c) The sparsity level; this is always required in one way or another in any sparsity promoting algorithm.

5. NUMERICAL EXAMPLES

In this section, the performance of the proposed algorithm is validated, within the system identification task. In the first experiment, we evaluate the performance of the proposed algorithm, in a non-distributed scenario, since, to our knowledge, sparsity-promoting adaptive algorithms, suitable for learning in diffusion networks, have not been proposed in the literature before. To be more specific, the proposed algorithm, is compared with the Adaptive Projection based algorithm using Weighted ℓ_1 Balls (APWL1) [1], with the Proportionate Adaptive Projection based onto Hyperslabs algorithm (PAPH), i.e., the proposed if we let $P^{(\boldsymbol{G}_n)}_{B_{\ell_1}[\boldsymbol{w}_n,\rho]} = I$, where I is the identity mapping, with the Online Cyclic Coordinate Descent Time Weighted Lasso (OCCD-TWL), the Online Cyclic Coordinate Descent Time and Norm Weighted LASSO (OCCD-TNWL), both proposed in [2], and with the LMS-based, Sparse Adaptive Orthogonal Matching Pursuit (Spadomp) [12]. The unknown vector is of dimension $m = 512$ and also $\|\boldsymbol{h}_*\|_0 = 20$. The coefficients of the input $\boldsymbol{u}_n = [u_n,\ldots,u_{n-m+1}]^T$ are drawn from a Gaussian distribution, with zero mean and standard deviation equal to 1. The noise process is Gaussian with variance equal to $\sigma^2 = 0.01$. Finally, the adopted performance metric is the average Mean Square Deviation (MSD), given by $\mathrm{MSD}(n) = 1/N\sum_{k=1}^{N}\|\boldsymbol{h}_{k,n} - \boldsymbol{h}_*\|^2$. In the projection-based algorithms, we choose $q = 55$, $\mu_n = 0.2\times\mathcal{M}_n$ and the width of the hyperslabs equals to $\epsilon = 1.3\times\sigma$. It should be pointed out that the performance of the algorithm turns out to be relatively insensitive to different choices of this parameter. A detailed experimental analysis on how different choices of ϵ affect the projection-based algorithms, has taken place in [1]. The radius of the weighted ℓ_1 ball, $\rho = \|\boldsymbol{h}_*\|_0$, whereas $\boldsymbol{w}_n$ and $\boldsymbol{G}_n^{-1}$ are constructed according to the strategy presented in Section 2 and the parameters are updated at every time instance, i.e., $n' = 1$. It should be stressed out that we experimentally observed that the proposed algorithm is rather insensitive to overestimated values of the sparsity level, which implies that even if we do not know the exact value of $\|\boldsymbol{h}_*\|_0$, if we set $\rho \geq \|\boldsymbol{h}_*\|_0$, the proposed algorithm exhibits a good performance; this behaviour was also observed in [1]. Regarding the parameter α, we observed that a value close to 1 leads to a fast convergence speed but it increases the steady state error floor, and vice versa. So, at the beginning of the adaptation, we choose $\alpha = 0.99$ and at every 250 time instances, we set $\alpha = \alpha/2$. In the OCCD-TWL and the OCCD-TNWL, the regularization parameter is chosen to be $\lambda_{\mathrm{TWL}} = \sqrt{2\sigma^2 n\mathrm{log}m}$, $\lambda_{\mathrm{TNWL}} = \sqrt{2\sigma^2 n^{4/3}\mathrm{log}m}$, respectively, as advised in [2]. The step size, adopted in the Spadomp, equals to 0.2, since this choice results in error floor similar to that of the algorithms, which employ projections onto weighted ℓ_1 balls. The forgetting factor of OCCD-TWN, OCCD-TNWL and Spadomp is set equal to 1 since, in the specific example, the system under consideration does not change with time. From Fig. 3, it can be seen that the proposed algorithm outperforms the APWL1, as it converges faster to the common error floor. This is due to the use of the variable metric projection. Nevertheless, the proposed and the APWL1 outperform significantly the PAPH. This implies that when seeking for sparse vectors, the projections onto the weighted ℓ_1 balls improve the performance of the respective algorithms.[2] Moreover, the proposed algorithm converges faster and the steady state error floor is slightly better compared to the Spadomp. We should point out, that the complexity of the Spadomp is $O(m)$, which implies that for the previously mentioned choice of q, the proposed algorithm is of higher complexity, albeit still of linear dependence on the number of free parameters. Compared to the OCCD-TWL, we observe that the performance of the proposed algorithm is slightly worse, yet the complexity of OCCD-TWL is much higher, which, in the absence of the shift invariance property of the input data, as it is commonly the case in sparse signal reconstruction applications, amounts to $O(m^2)$. Finally, the OCCD-TNWL outerforms the other algorithms, at the expense of a higher complexity, which is approximately twice than that of OCCD-TWL.

In the second experiment, we consider a diffusion network consisted of $N = 10$ nodes; the unknown vector is of dimension $m = 256$, and the number of non-zero coefficients equals to 20. The input, at each node, is chosen as in the previous experiment, the variance of the noise equals to $\sigma_k^2 = 0.01\varsigma_k\ \forall k \in \mathcal{N}$, where $\varsigma_k \in [0.5, 1]$ is randomly chosen according to the uniform distribution, and the combination coefficients are chosen with respect to the Metropolis

[2] This trend, i.e., the improved steady state error floor of the algorithms, which employ projections onto weighted ℓ_1 balls, compared to the proportionate ones, was also verified, via an extensive comparative study with the main representatives of the of the proportionate algorithmic family.

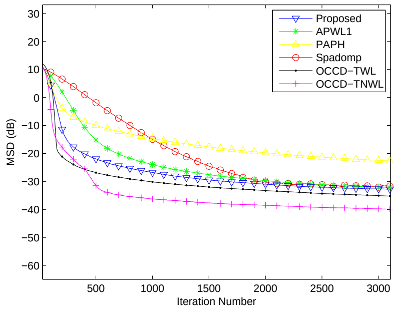


Fig. 3. Identification of a sparse system in a non-distributed experiment.

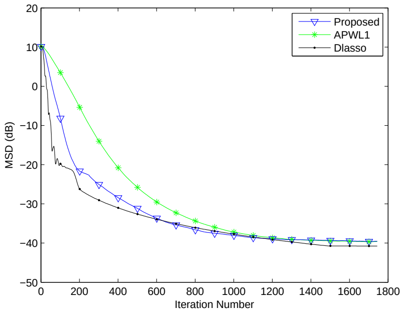


Fig. 4. Identification of a sparse system in a distributed experiment.

rule [3]. The proposed algorithm is compared with the distributed APWL1, i.e., if we let $\boldsymbol{G}_n = \boldsymbol{I}_m$, and the distributed Lasso (Dlasso) [5]. The Dlasso is a batch algorithm, so, at every time instance that a new pair of data becomes available, the algorithm is re-initialized so as to solve a new optimization problem. In the projection-based algorithms, $q = 20$ and the rest of the parameters remain the same as in the previous experiment. Finally, in the Dlasso, the regularization parameter is chosen equal to $\|\boldsymbol{h}_*\|_1$. From Fig. 4, it can be seen that the proposed algorithm outperforms the APWL1, and it exhibits a faster convergence speed. The Dlasso outperforms the projection-based algorithms only slightly, albeit an inversion of a matrix of dimensions proportionate to m has to be performed at each time instant. The reported performance corresponds to the best obtained one, after extensive experimentation with respect to the involved parameters. It is interesting to note that in the case of the distributed LASSO, the comparative performance gains of the LASSO in Fig. 3, seem to be lost.

In the third experiment, we study the robustness of the proposed scheme, with respect to adopting different strategies in order to construct $\boldsymbol{w}_n$ and $\boldsymbol{G}_n$. To this end, we consider the following strategies: a) the previously mentioned quantities are constructed via the node with the smallest noise variance (Proposed a), b) $\boldsymbol{w}_n$ and $\boldsymbol{G}_n$ are generated by the node with the largest variance (Proposed b) and c) $\boldsymbol{w}_n$ and $\boldsymbol{G}_n$ are constructed locally at every node (Proposed c). Note that the latter one violates the theoretical assumption of having common weights to all nodes. In order to verify whether the nodes reach consensus, instead of presenting the average MSD, as in the previous experiments, we fix two arbitrary nodes (node 1 and node 5), and we plot the local MSD curves, for each one of the previously mentioned strategies. From Fig. 5, it can be seen that the performance of the proposed algorithm is not affected by the strategy used in order to construct $\boldsymbol{w}_n$ and $\boldsymbol{G}_n$, and, finally, the steady state error floors nearly coincide, which implies that the nodes converge to estimates, which are almost the same.

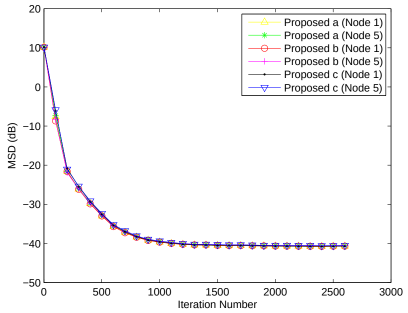


Fig. 5. Performance of the algorithm with respect to the strategy adopted in order to construct $\boldsymbol{w}_n$ and $\boldsymbol{G}_n$.

6. REFERENCES

[1] Y. Kopsinis, K. Slavakis, and S. Theodoridis, "Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls," *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 936 –952, 2011.

[2] D. Angelosante, J.A. Bazerque, and G.B. Giannakis, "Online adaptive estimation of sparse signals: where RLS meets the ℓ_1-norm," *IEEE Transactions on Signal Processing,*, vol. 58, no. 7, pp. 3436–3447, 2010.

[3] C.G. Lopes and A.H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

[4] S. Chouvardas, K. Slavakis, and S. Theodoridis, "Adaptive robust distributed learning in diffusion sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4692 –4707, 2011.

[5] G. Mateos, J.A. Bazerque, and G.B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.

[6] F.S. Cattivelli and A.H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035 –1048, 2010.

[7] G. Mateos, I.D. Schizas, and G.B. Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, 2009.

[8] M. Yukawa and I. Yamada, "A unified view of adaptive variable-metric projection algorithms," *EURASIP Journal on Advances in Signal Processing*, vol. 2009.

[9] J. Benesty and S.L. Gay, "An improved PNLMS algorithm," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002, pp. 1881–1884.

[10] S. Werner, J.A. Apolinário Jr, and P.S.R. Diniz, "Set-membership proportionate affine projection algorithms," *EURASIP J. Audio, Speech, and Music Processing*, vol. 2007, no. 1, pp. 1–10, 2007.

[11] S. Werner, Y.F. Huang, M.L.R. De Campos, and V. Koivunen, "Distributed parameter estimation with selective cooperation," in *ICASSP*. IEEE, 2009, pp. 2849–2852.

[12] G. Mileounis, B. Babadi, N. Kalouptsidis, and V. Tarokh, "An adaptive greedy algorithm with application to nonlinear communications," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 2998–3007, 2010.