EUSIPCO 2013 1569744467

TRAINING–BASED AND BLIND ALGORITHMS FOR SPARSITY–AWARE DISTRIBUTED LEARNING

Symeon Chouvardas[#], *Gerasimos Mileounis*[*], *Nicholas Kalouptsidis*[#] *and Sergios Theodoridis*[#]

[#]Dept. of Informatics and Telecommunications, University of Athens, Ilisia 15784, Greece
[*]Accenture Global Talent Innovation Network (GTIN)–Greece

ABSTRACT

In this paper, two novel algorithms for distributed estimation of sparse signals are presented. The algorithms follow an iterative greedy two–step procedure. The first algorithm operates in a training based manner, *i.e.*, the nodes of the network have access to input–output data, whereas the second operates blindly, *i.e.*, the nodes observe output data only. In both cases, the nodes cooperate with each other, by exchanging information with the neighboring nodes. The goal is twofold, first to identify the support set of the unknown signal, and then the non–zero values, which are restricted in the active support set. Theoretical results are outlined and an experimental validation of the proposed algorithms is carried out.

Index Terms— Distributed systems, compressed sensing, system identification, greedy algorithms

1. INTRODUCTION

Many real–life signals and systems adhere to parsimonious models. In other words, they comprise a small number of significant coefficients, whereas the rest are either zero or have negligible amplitudes. Typical examples of sparse signals/systems are: wireless multipath channels, acoustic signals, seismic data, image deblurring and High Definition TV [1].

There are two major algorithmic approaches to sparsity–aware learning. The first promotes sparsity by embedding into the optimization problem the ℓ_1 norm constraint, *e.g.*, [2–4]. It is by now well established that such a constraint promotes sparse solutions. The other approach relies on the greedy viewpoint, [5, 6]. Greedy techniques identify the positions, in which the non–zero coefficients lie (known as *support set*), and then restrict the estimation step in this subset.

With only few exceptions, *e.g.*, [7–10], sparsity promoting algorithms assume that the training data, through which the unknown target vector is estimated, are centrally available. That is, there exist a central processing unit (or Fusion Center, FC), which has access to all the training data, and performs all essential computations. Nevertheless, in many applications the need for decentralized/distributed processing rises. Typical examples of such applications are those involving wireless sensor networks (WSNs), which are deployed over a geographical region, and the nodes are tasked to collaboratively estimate an unknown (but common) sparse vector. The existence of a fusion center, which collects all training data and then performs the required computations, may be prohibited due to geographical constraints and/or energy, bandwidth limitations. Henceforth, the development of distributed algorithms is of significant importance.

A scarce design factor in data networks is the available bandwidth and hence needs to be managed carefully. Therefore, it is desirable for each node to obtain the minimum training data without consuming much bandwidth. Besides, the existence of a training sequence can impose a significant overhead cost. Blind identification methods rely only on output data to identify the sparse signal and hence eliminate the need for training [11]. Therefore blind identification algorithms offer a bandwidth efficient solution to distributed learning. The basic tools to explore blind identification strategies are maximum likelihood methods, subspace techniques and statistical methods, using either second order or higher order information [11],[12]. In this work, Higher Order Statistical (HOS) information from the available output data of each node is used cooperatively to estimate the unknown vector.

In this paper, two novel algorithms for sparse signal estimation in distributed networks are proposed. The first is appropriate for training–based operation. Each node has access to a finite number of input–output measurements. Besides their own measurements, network nodes exploit information, which comes from their neighbours, where communication is possible. The training–based algorithm starts with data fusion under a certain protocol; next, the nodes individually compute their own support set. The s dominant positions of the unknown vector are recovered and then a Least Squares step, restricted on this subset, follows. A theoretical analysis is sketched and experimental results highlight the enhanced performance of the proposed batch scheme, compared to an

Email: {schouv,gmil,kalou,stheodor}@di.uoa.gr.

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: Thales. Investing in knowledge society through the European Social Fund.

existing sparsity promoting distributed algorithm. The greedy operating mode is subsequently employed in the blind setup. The resulting algorithm exploits network information to enhance the sample statistics in each node, in order to perform recovery of the unknown sparse signal.

Notation: Vectors will be denoted by boldfaced letters and matrices by uppercase boldface letters. The symbol $(\cdot)^T$ will stand for the transpose. Moreover, the set of all real numbers will be denoted by $\mathbb{R}$. For a vector $\mathbf{x} := [x_1, \ldots, x_m]^T \in \mathbb{R}^m$, the term $\mathrm{supp_s}(\mathbf{x})$, denotes its support set, which contains the s largest in amplitude positions of $\mathbf{x}$. Given a certain set $\mathcal{S}$, its cardinality will be denoted by $|\mathcal{S}|$. Finally, $\|\mathbf{x}\|_{\ell_2}$ will be the Euclidean norm and $\|\mathbf{x}\|_{\ell_1} := \sum_{i=1}^{m} |x_i|$ the ℓ_1 norm of $\mathbf{x}$.

2. SPARSE PROBLEM FORMULATION

Our task is to estimate an unknown sparse parameter vector $\boldsymbol{h}_* \in \mathbb{R}^m$, exploiting a finite number of measurements collected at the N nodes of an ad–hoc network. 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 *neighborhood* of k. The input–output relation adheres to the following linear model:

$$\boldsymbol{y}_k = \boldsymbol{A}_k \boldsymbol{h}_* + \boldsymbol{\eta}_k, \ \forall k \in \mathcal{N} \tag{1}$$

where $\boldsymbol{A}_k$ is an $l \times m$ sensing matrix, with $l \ll m$, $\boldsymbol{y}_k \in \mathbb{R}^l$ and $\boldsymbol{\eta}_k \in \mathbb{R}^l$ is the noise process. The vector to be estimated is assumed to be at most s–sparse, *i.e.*, $\|\boldsymbol{h}_*\|_{\ell_0} \leq s \ll m$, where $\|\cdot\|_{\ell_0}$ denotes the ℓ_0 quasi–norm.

In non–distributed sparsity–aware learning the goal is the estimation of the vector $\boldsymbol{h}_*$, using fewer measurements, l, than the dimension of the problem m. A representative example of a sparsity promoting algorithm is the Least Absolute Shrinkage Selection Operator (LASSO), which solves the following optimization problem

$$\hat{\boldsymbol{h}} = \arg \min_{\|\boldsymbol{h}\|_{\ell_1} \leq \delta} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}\|_{\ell_2}^2, \tag{2}$$

where the node subscript k is surpressed, and δ is a user–defined radius of the ℓ_1–norm. Notice that the term $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h}\|_{\ell_2}^2$ accounts for the misfit between the input–output measurements, and the ℓ_1–norm constraint promotes sparsity by shrinking small coefficients towards zero. Several techniques have been proposed for solving the optimization problem described in (2), *e.g.*, [3].

We turn now our focus on the distributed sparsity–aware estimation task and rewrite (2) as follows:

$$\hat{\boldsymbol{h}} = \arg \min_{\|\boldsymbol{h}\|_{\ell_1} \leq \delta} \sum_{k \in \mathcal{N}} \|\boldsymbol{y}_k - \boldsymbol{A}_k \boldsymbol{h}\|_{\ell_2}^2. \tag{3}$$

This problem can be reformulated in a fully decentralized way, where each node exploits local information, as well as information received by the neighborhood. The decentralized optimization can be efficiently solved by adopting the Alternating Direction–Method of Multipliers (AD–MoM), [13]. It has been shown in [7], that the solution, which occurs if ones solves the decentralized problem, coincides with the solution obtained by solving (3). It should be pointed out that each node lacks global network information; however if an appropriate cooperation protocol is adopted, then the global solution can be obtained.

3. GREEDY TECHNIQUES AND THE PROPOSED TRAINING–BASED ALGORITHM

Greedy techniques under the centralized scenario, iteratively estimate the unknown parameter by applying the following two–step approach:

- **Subset Selection**: After the computation of the proxy signal, which is constructed based on input–output measurements, the s–dominant terms are computed. These comprise the identified support set.
- **Greedy Update**: The estimate of the unknown vector is computed, by performing a Least–Squares restricted on the identified support set.

The performance of the greedy–based algorithms is crucially affected by the choice of the proxy signal. One of the first approaches is the Orthogonal Matching Pursuit (OMP) [14]. The OMP proxy signal equals to $\boldsymbol{p} = \boldsymbol{A}^T(\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{h}}) \approx \boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{h}_* - \hat{\boldsymbol{h}})$, where $\hat{\boldsymbol{h}}$ stands for the most recent estimate. At each iteration, the largest coefficient of the proxy is computed, and is added on the current estimate of the support set. This procedure is repeated s times and the support set is estimated. Noise sensitivity is a major drawback of this algorithm. A more sophisticated approach proposed in [5], is the Compressed Sampling Matching Pursuit (CoSaMP). This algorithm allows to chose multiple indices at each iteration and results to a better performance compared to the OMP. An adaptive version of the CoSaMP, known as SpAdOMP has been proposed in [6]. A different approach, presented in [15], selects the s dominant coefficients of the following proxy: $\boldsymbol{p} = \hat{\boldsymbol{h}} + \boldsymbol{A}^T(\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{h}}) \approx \boldsymbol{h}_*$. This idea will be followed in this study, since it leads to enhanced performance compared to other schemes and allows network support set consensus, *i.e.*, the nodes will agree on the same support set in the distributed scenario, due to its non time varying nature.

In distributed greedy–based algorithms, information is exchanged, so as to achieve the following goals:

- **Support set Consensus.** That is, all network nodes identify the same support set.
- **Information enhancement.** In scenarios where the number of measurements is small, exchanging data with the neighborhood and fusing them under a certain protocol, can lead to better results.

The steps of the proposed algorithm are summarized in Table 1. In steps 1 and 2, the nodes exchange their input–output

Table 1. The Distributed Hard Thresholding pursuit Algorithm (DiHaT)

Algorithm description		Complexity
$\boldsymbol{h}_{k,0}=\mathbf{0}_m, \quad \overline{\boldsymbol{y}}_{k,0}=\boldsymbol{y}_k, \quad \overline{\boldsymbol{A}}_{k,0}=\boldsymbol{A}_k \quad \mathcal{S}_{k,0}=\emptyset$	{Initialization}	
Loop		
1: $\overline{\boldsymbol{y}}_{k,n}=\sum_{r\in\mathcal{N}_k} a_{r,k}\overline{\boldsymbol{y}}_{r,n-1}$	{Combine local output measurements}	$\mathcal{O}(\lvert\mathcal{N}_k\rvert l)$
2: $\overline{\boldsymbol{A}}_{k,n}=\sum_{r\in\mathcal{N}_k} a_{r,k}\overline{\boldsymbol{A}}_{r,n-1}$	{Combine local input measurements}	$\mathcal{O}(\lvert\mathcal{N}_k\rvert ml)$
3: $\mathcal{S}_{k,n}=\text{supp}_s\left(\boldsymbol{h}_{k,n-1}+\overline{\boldsymbol{A}}_{k,n}^T(\overline{\boldsymbol{y}}_{k,n}-\overline{\boldsymbol{A}}_{k,n}\boldsymbol{h}_{k,n-1})\right)$	{Identify s largest components}	$\mathcal{O}(m^2 l)$
4: $\hat{\boldsymbol{h}}_{k,n}=\arg\min_{\boldsymbol{h}}\left\{\lVert\overline{\boldsymbol{y}}_{k,n}-\overline{\boldsymbol{A}}_{k,n}\boldsymbol{h}\rVert_{\ell_2}^2\right\}, \text{supp}(\boldsymbol{h})\subseteq\mathcal{S}_{k,n}$	{Signal Estimation}	$\mathcal{O}(ms)$
5: $\tilde{\boldsymbol{h}}_{k,n}=\sum_{r\in\mathcal{N}_k} b_{r,k}\hat{\boldsymbol{h}}_{r,n}$	{Combine local estimates}	$\mathcal{O}(\lvert\mathcal{N}_k\rvert m)$
6: $\boldsymbol{h}_{k,n}=\text{supp}_s(\tilde{\boldsymbol{h}}_{k,n})$	{Identify the s largest components of the estimate}	$\mathcal{O}(m)$
Until halting condition is *true*		

measurements and fuse them under a certain protocol. This information fusion is dictated by the combination weights $a_{r,k},\ \forall k\in\mathcal{N},\ \forall r\in\mathcal{N}_k$. As it will become clear later on, if these coefficients are chosen properly, then $\overline{\boldsymbol{A}}_{k,n}$ and $\overline{\boldsymbol{y}}_{k,n}$ tend asymptotically to the average values, *i.e.,* $\frac{1}{N}\sum_{r=1}^N \boldsymbol{A}_r$ and $\frac{1}{N}\sum_{r=1}^N \boldsymbol{y}_r$, respectively. This significantly improves the performance of the algorithm, since as the number of iteration increases, the data available at each node are enhanced, in the sense that the proxy selection and the parameter estimation procedure will contain information which comes from the entire network. In step 3, the s largest in amplitude coefficients of the signal proxy are selected, and step 4 performs a Least–Squares operation in the support set computed in the previous step. In step 5, the nodes exchange their estimates and fuse them in a similar way as in steps 1,2. Finally, since the nodes have access to different measurements, especially in the first iterations in which their input–output data are not close to the average values, the estimated support sets among the neighborhood may be different. This implies that in step 5 there is no guarantee that the produced estimate will be s–sparse. To this end, in step 6 a thresholding operation takes place and the final estimate at each node is s–sparse.

3.1. Convergence Analysis

Let us define the $N\times N$ combination matrix $\boldsymbol{W}_1$ with entries given by $a_{r,k}$. Following [16], $\boldsymbol{W}_1$ is chosen so that it possesses the following properties:

1. $\mathbf{1}_N^T\boldsymbol{W}_1=\mathbf{1}_N^T$, where $\mathbf{1}_N\in\mathbb{R}^N$ is the vector of ones.
2. $\boldsymbol{W}_1\mathbf{1}_N=\mathbf{1}_N$.
3. $\lambda(\boldsymbol{W}_1-\mathbf{1}_N\mathbf{1}_N^T/N)<1$, where $\lambda(\cdot)$ stands for the maximum eigenvalue of the respective matrix.

The same assumptions also hold for the matrix $\boldsymbol{W}_2$ with entries $b_{r,k}$. Methodologies for constructing the combination matrix, so as to fulfil the previously mentioned assumptions in a decentralized fashion, have been proposed in [16].

The noise term $\boldsymbol{\eta}_k$ follows the Gaussian distribution $N(\mathbf{0}_l,\sigma_k^2\boldsymbol{I}_l),\ \forall k\in\mathcal{N}$, where σ_k^2 is the noise variance at the k–th node, which is assumed to be bounded by a finite constant $\forall k\in\mathcal{N}$, and $\mathbf{0}_l,\boldsymbol{I}_l$ are the $l\times 1$ vector of zeros and the $l\times l$ identity matrix respectively. Moreover, the noise terms are spatially independent over the nodes.

Since the vectors $\boldsymbol{\eta}_k$ are independent over the nodes, and their variance is bounded, the strong law of large numbers ensures that $(1/N)\sum_{r=1}^N\boldsymbol{\eta}_r,\ N\to\infty$ converges almost surely to its expected value which is zero. Furthermore, the assumptions imposed on $\boldsymbol{W}_1$ in conjunction with the results of [16, 17], imply that $\overline{\boldsymbol{y}}_{k,n}$ converges to its mean $\lim_{n\to\infty}\overline{\boldsymbol{y}}_{k,n}=(1/N)\sum_{r=1}^N\boldsymbol{y}_r$. We replace $\boldsymbol{y}_k$ from (1) to obtain $\lim_{n\to\infty}\overline{\boldsymbol{y}}_{k,n}=((1/N)\sum_{r=1}^N\boldsymbol{A}_r)\boldsymbol{h}_*+(1/N)\sum_{r=1}^N\boldsymbol{\eta}_r$ or $\lim_{n\to\infty}\overline{\boldsymbol{y}}_{k,n}\approx\lim_{n\to\infty}\overline{\boldsymbol{A}}_{k,n}\boldsymbol{h}_*$. The last relation holds due to the properties of $\boldsymbol{W}_1$, and the fact that $(1/N)\sum_{r=1}^N\boldsymbol{A}_r$ is the limit of the sequence defined in step 2 of table 1. For the derivation of the Theorem we will make the approximation that after a large number of iterations, n_0, the previously mentioned limits hold, *i.e.,* $\overline{\boldsymbol{y}}_{k,n}\approx((1/N)\sum_{r=1}^N\boldsymbol{A}_r)\boldsymbol{h}_*\approx\overline{\boldsymbol{A}}_{k,n}\boldsymbol{h}_*,\ \forall k\in\mathcal{N},\ \forall n\geq n_0$.

Theorem 1. *Under the previous discussion and a restricted isometry assumption with constant $\delta_{3s}<\frac{1}{3}$, $\forall k\in\mathcal{N}$, for the matrix $(1/N)\sum_{r=1}^N\boldsymbol{A}_r$, it holds that*

$$\lVert\underline{\boldsymbol{h}}_{n+1}-\underline{\boldsymbol{h}}_*\rVert_{\ell_2}\leq\rho\lVert\underline{\boldsymbol{h}}_n-\underline{\boldsymbol{h}}_*\rVert_{\ell_2}, \tag{4}$$

where $\rho=\max_{k\in\mathcal{N}}\left\{\sqrt{\frac{8\delta_{3s}^2}{1-\delta_{2s}^2}}\right\}<1$, $\underline{\boldsymbol{h}}_n=[\boldsymbol{h}_{1,n}^T,\ldots,\boldsymbol{h}_{N,n}^T]^T\in\mathbb{R}^{Nm}$ *and* $\underline{\boldsymbol{h}}_*=[\boldsymbol{h}_*^T,\ldots,\boldsymbol{h}_*^T]^T\in\mathbb{R}^{Nm}$.

Proof: The proof will be presented in a forthcoming expanded version of this work.

3.2. Blind Diffusion Pursuit Algorithm

This section addresses the problem of sparse blind identification in distributed networks. The proposed algorithm relies on a distributed one step greedy scheme and relies on higher order cumulants.

Table 2. The BlinD distributed Pursuit Algorithm

Algorithm description	Complexity
While $i <$ Consensus Iterations	
1: $\hat{\bar{\boldsymbol{c}}}_{py}^{(k,i)}(m,\boldsymbol{\tau}) = \sum_{r\in\mathcal{N}_k} a_{r,k}\bar{\boldsymbol{c}}_{py}^{(r,i-1)}(m,\boldsymbol{\tau})$	$\mathcal{O}(\lvert\mathcal{N}_k\rvert m)$
2: $i = i+1$	
End While	
3: $\mathcal{S}_k = \text{supp}_s\left(\hat{\bar{\boldsymbol{c}}}_{py}^{(k,i)}(m,\boldsymbol{\tau})\right)$	$\mathcal{O}(m)$
While $i <$ Consensus Iterations	
4: $\widehat{\boldsymbol{C}}_{\mathcal{S}_k\lvert\mathcal{S}_k}^{(k,i)} = \sum_{r\in\mathcal{N}_k} b_{r,k}\boldsymbol{C}_{\mathcal{S}_r\lvert\mathcal{S}_r}^{(r,i-1)}$	$\mathcal{O}(\lvert\mathcal{N}_k\rvert s^2)$
5: $i = i+1$	
End While	
6: $\widehat{\boldsymbol{C}}_{\mathcal{S}_k\lvert\mathcal{S}_k}^{(k,i)} = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T$	$\mathcal{O}(s^3)$
7: $\widehat{\boldsymbol{h}}_{k\lvert\mathcal{S}_k} = u_{1,1}\sigma_{1,1}\boldsymbol{v}_{1:s,1}, \quad \widehat{\boldsymbol{h}}_{\lvert\mathcal{S}_k^C} = \boldsymbol{0}$	

In a centralized operating mode, an one step greedy algorithm is presented in [18]. Following similar rationale as in [18], we consider the system:

$$y_{k,n} = \boldsymbol{a}_{k,n}^T\boldsymbol{h}_* + \eta_{k,n}, \forall k\in\mathcal{N},\ n = 1,\ldots,l, \qquad (5)$$

where $y_{k,n}$ is the output, $\boldsymbol{a}_{k,n} = [a_{k,n},\ldots,a_{k,n-m+1}]^T$ is the input and $\eta_{k,n}$ is the noise process. The previous data generation model can be written compactly in a matrix–vector form as in (1). The pth order output cumulant of Eq. (5), (the single system case) when driven by a stationary independent and identically distributed white noise process, is given by the Brillinger–Roseblatt formula:

$$c_{py}(\tau_1,\ldots,\tau_{p-1}) = \gamma_{px}\sum_{i=0}^{m} h_{*,i}\prod_{j=1}^{p-1} h_{*,i+\tau_j}, \qquad (6)$$

where $h_{*,i}$ is the i–th coefficient of the vector $\boldsymbol{h}_*$ and γ_{px} is the pth order cumulant of the input with $c_{px}(\tau_1,\ldots,\tau_{p-1}) = \gamma_{px}\delta(\tau_1,\ldots,\tau_{p-1})$. To avoid the inherent scaling ambiguities that exist in blind identification algorithms, it is assumed that: (a) $h_{*,0} = 1$, (b) $\gamma_{px}\neq 0$ and (c) $h_{*,m}\neq 0$. The support set of the unknown signal is chosen using output cumulant information at properly selected lags of the form

$$\begin{aligned}\bar{c}_{py}(m,\tau) &= c_{py}(m,\tau,0,\ldots,0) = \gamma_{px}h_{*,m}h_{*,\tau}h_{*,0}^{p-2}\\ &= \gamma_{px}h_{*,m}h_{*,\tau},\ \text{with}\ \tau = 0,\ldots,m. \qquad (7)\end{aligned}$$

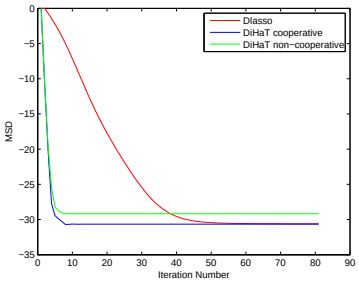
Therefore collecting all cumulant information of Eq. (7) in a vector, $\bar{\boldsymbol{c}}_{py}(m,\boldsymbol{\tau})$, gives rise to the cumulant proxy where its dominant in magnitude positions reflect the support set of $\boldsymbol{h}_*$ [18]. The next step is to restrict the estimation step to the identified support set, $\mathcal{S}$. The estimation method can be any of the ones reported in [12, Chapter 7]. Here we adopt the m–SVD which is found in Table 2 steps 4–5 whose ijth element of $\widehat{\boldsymbol{C}}_{\mathcal{S}_k\lvert\mathcal{S}_k}^{(k,0)} = c_{4y}(m,i,j)$ with $0\leq i,j\leq m$ [18].

A weakness of the above algorithm, is that in practice the true cumulants are replaced by sample cumulants, denoted by $\hat{\boldsymbol{c}}_{py}(\tau_1,\ldots,\tau_{p-1})$. Sample cumulants are estimated from output observations of finite length, which introduce bias/variance distortion. A well known techniques to tackle bias/variance distortion is the segmentation of the observed data into non–overlapping records. More specifically, sample cumulant estimates are obtained from each segment, and then averaged across the segments to obtain estimates of low bias/variance. Such an operating mode arises naturally in distributed networks, where each node has its own measurements and can exchange/combine them with other nodes. Therefore, the first algorithm of [18] is next modified to operate in a distributed fashion, where cooperation helps it reduce bias/variance problems.

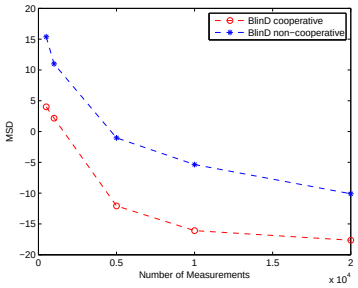
The algorithm outlined in Table 2 consists of four major steps. In the first step, it runs distributed averaging consensus iterations over the network, *i.e.* computes the average of the sample cumulant proxy vector, given at the nodes (see Step 1). Distributed averaging is performed as outlined in section 3.1. More specifically, for a user–defined number of iterations, the nodes exchange information with the neighborhood and fuse them using the combination weights. Afterwards, the network comes approximately to consensus or agreement to a common cumulant proxy vector. Then in Step 3., each node performs hard thresholding to the s–largest indices of the cumulant proxy vector in order to identify the support set, $\mathcal{S}_k$, of the unknown signal. These two steps constitute the subset selection mechanism for the outlined algorithm. Next, comes the greedy update and as before (Step 4.) a distributed averaging iteration is performed to the cumulant matrix restricted on the identified subset $\boldsymbol{C}_{\mathcal{S}_k\lvert\mathcal{S}_k}^{(k,i)}$, which is required for identification. Finally, the last two steps estimate the unknown signal using the techniques outlined above.

4. COMPUTER SIMULATIONS

In the first experiment, the DiHaT is compared to the Dlasso proposed in [7]. Moreover, the performance of the DiHaT is validated in a scenario where the nodes do not cooperate with each other, or in other words they produce estimates relying solely on their local input–output measurements. A network with $N = 20$ nodes is considered, the dimension of the unknown vector equals to $m = 70$ and the number of measurements at each node is $l = 55$. Furthermore, for the unknown vector we have that $\|\boldsymbol{h}_*\|_{\ell_0} = 10$. The input matrix $\boldsymbol{A}_k$ follows the Gaussian distribution with zero mean and variance 1. Moreover, the noise is generated with respect to the Gaussian distribution and a Signal to Noise Ratio (SNR) at each node equal to 20 dB. The combination coefficients $a_{r,k},\ b_{r,k}$ are selectedby the Metropolis rule [16]. The performance metric which is presented is the average normalized Mean Square Deviation which equals to $\text{MSD}(n) = \frac{1}{N}\sum_{k\in\mathcal{N}}\frac{\|\boldsymbol{h}_{k,n}-\boldsymbol{h}_*\|_{\ell_2}^2}{\|\boldsymbol{h}_*\|_{\ell_2}^2}$ and the curves result from an averaging of 100 independent Monte Carlo runs. The first computer experiment tests the proposed training based method versus



(a) MSD for the first experiment.



(c) MSD for the second experiment.

Fig. 1. Average MSD Performance of the proposed algorithms

Dlasso. The regularization parameter, which is user defined in the Dlasso, is computed via a cross validation procedure, as proposed in [7]. Fig. 1.a illustrates that the DiHaT outperforms the Dlasso, in the sense that it converges faster to a similar error floor. Furthermore, the DiHaT in the non-cooperative scenario converges to a higher error floor, which indicates that the cooperation among the nodes enhances the results. It should be pointed out that the Least Squares operation of the DiHaT takes place in the identified support set, which reduces significantly the dimensionality, in contrast to the Dlasso, where all the operations take place in the original space of dimension m.

In the second experiment, the performance of the BlinD distributed pursuit is examined. To this end, the performance of the proposed blind scheme is compared to the non–cooperative scenario. The network consists of $N = 20$ nodes, $m = 50$, $s = 5$ and the SNR equals 20 dB. The input is an independent and identically distributed QPSK signal. The number of consensus iterations at each node is 20. Fig 1.c illustrates that the cooperation among the nodes improves the performance of the algorithm, since a better MSD can be obtained even if the number of measurements is small, at the expense of the extra computational complexity coming from the consensus iterations.

5. CONCLUSION

In this paper, two novel algorithms for distributed sparse vector estimation are proposed. The first is suitable for training based operation, whereas the second operates blindly. Both algorithms follow the greedy principle. Theoretical results of the training based scheme are discussed, and the performance of the algorithms is validated through experiments. Future research will be focused on sparse blind algorithms for non-linear systems.

6. REFERENCES

[1] Alfred M Bruckstein, David L Donoho, and Michael Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, vol. 51, no. 1, pp. 34–81, 2009.

[2] Emmanuel J Candès and Michael B Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[3] Sergios Theodoridis, Yannis Kopsinis, and Konstantinos Slavakis, "Sparsity–aware learning and compressed sensing: An overview," *arXiv preprint arXiv:1211.5231*, 2012.

[4] 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.

[5] D. Needell and J.A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301 – 321, 2009.

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

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

[8] João FC Mota, João MF Xavier, Pedro MQ Aguiar, and Markus Püschel, "Distributed basis pursuit," *arXiv preprint arXiv:1009.1128*, 2010.

[9] Paolo Di Lorenzo and Ali H Sayed, "Sparse distributed learning based on diffusion adaptation," *arXiv preprint arXiv:1206.3099*, 2012.

[10] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis, "A sparsity promoting adaptive algorithm for distributed learning," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5412 –5425, oct. 2012.

[11] Z. Ding and Y. Li, *Blind Equalization and Identification (Signal Processing and Communications)*, CRC Press, 2001.

[12] C. Nikias and A.P. Petropulu, *Higher Order Spectra Analysis*, Prentice Hall, 1993.

[13] Dimitri P Bertsekas and John N Tsitsiklis, "Parallel and distributed computation: Numerical methods," 1999.

[14] Joel A Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[15] S. Foucart, "Hard thresholding pursuit: An algorithm for compressive sensing," *SIAM Journal on Numerical Analysis*, vol. 49, no. 6, pp. 2543–2563, 2011.

[16] Lin Xiao and Stephen Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[17] Lin Xiao, Stephen Boyd, and Sanjay Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *International Symposium on Information Processing in Sensor Networks IPSN*. IEEE, 2005, pp. 63–70.

[18] G. Mileounis and N. Kalouptsidis, "A sparsity driven approach to cumulant based identification and order determination," *Signal Processing (ELSEVIER)*, 2013.