Adaptive Robust Distributed Learning in Diffusion Sensor Networks

Symeon Chouvardas, *Student Member, IEEE*, Konstantinos Slavakis, *Member, IEEE*, and Sergios Theodoridis, *Fellow, IEEE*

***Abstract*—In this paper, the problem of adaptive distributed learning in diffusion networks is considered. The algorithms are developed within the convex set theoretic framework. More specifically, they are based on computationally simple geometric projections onto closed convex sets. The paper suggests a novel combine-project-adapt protocol for cooperation among the nodes of the network; such a protocol fits naturally with the philosophy that underlies the projection-based rationale. Moreover, the possibility that some of the nodes may fail is also considered and it is addressed by employing robust statistics loss functions. Such loss functions can easily be accommodated in the adopted algorithmic framework; all that is required from a loss function is convexity. Under some mild assumptions, the proposed algorithms enjoy monotonicity, asymptotic optimality, asymptotic consensus, strong convergence and linear complexity with respect to the number of unknown parameters. Finally, experiments in the context of the system-identification task verify the validity of the proposed algorithmic schemes, which are compared to other recent algorithms that have been developed for adaptive distributed learning.**

***Index Terms*—Adaptive filtering, adaptive projected subgradient method, consensus, distributed learning, diffusion networks.**

I. Introduction

DISTRIBUTED networks, in which nodes are tasked to collect information and estimate a parameter of interest, are envisioned to play a central role in many applications. Typical examples of these are: environmental monitoring, acoustic source localization, distributed classification, life sciences, etc., [1]–[5]. The general concept of such networks can be summarized as follows:

- the nodes sense an amount of data from the environment;
- the essential computations, in order to estimate the unknown parameter, are performed in each one of the nodes;
- each node transmits the obtained estimate to a subset of nodes, with which communication is possible, under the network's topology constraints.

The goal is to drive the estimates, that are obtained from all the nodes, to converge to the same value (consensus) of the unknown parameter. In such a way, one can exploit all the data information, which becomes available at each node.

A first approach to the problem is to resort to a centralized solution; each node collects data, which are subsequently transmitted to a central node, also known as the fusion center, in which the computations are performed. This philosophy is the simplest one, albeit it is not always feasible to be adopted, due to geographical constraints and/or limited bandwidth. Moreover, this scheme lacks robustness, since whenever the fusion center fails the whole network collapses. Henceforth, one has to seek for a decentralized solution, where the nodes can themselves take part in the computations.

A typical paradigm of a decentralized distributed network topology is the incremental one [6], [7]. Each sensor is able to communicate with only one node and consequently the nodes constitute a cyclic pattern. Although this topology requires small communications bandwidth, it is not practical to be applied in networks with many nodes. Moreover, in the case of a possible node failure, the network collapses. For these reasons, in many applications, a diffusion topology is adopted, where each sensor can communicate with a subset of nodes, which define its neighborhood. Each node receives information that consists of the updates of the unknown parameter, which have taken place in the nodes of its neighborhood. In the sequel, the steps of information fusion and parameter adaptation are performed according to a protocol. Although this topology requires larger bandwidth and the convergence analysis is more challenging, it accelerates the procedure of estimating the unknown parameter ([8]–[10]) compared to the case where each node works individually. Moreover, it is robust to cope with cases where a number of nodes are malfunctioning and its implementation is easier when large networks are involved.

The algorithms which are developed in this paper belong to the family of the adaptive projected subgradient methodology, [11]–[13], which enjoys a number of advantages such as the following:

- any form of continuous nonnegative convex loss function can be used, without altering the structure of the algorithm as well as its theoretical analysis;
- convex constraints can be readily incorporated in the optimization task, in a rather trivial way, e.g., [12], [13];
- the computations are cast in terms of inner products, hence the *kernel trick* can potentially be exploited in order to deal

Manuscript received November 09, 2010; revised March 16, 2011 and June 07, 2011; accepted June 18, 2011. Date of publication July 12, 2011; date of cur- rent version September 14, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Hideaki Sakai. This work was supported in part 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.

S. Chouvardas and S. Theodoridis are with the Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilissia, Athens 157 84, Greece (e-mail: schouv@di.uoa.gr; stheodor@di.uoa.gr).

K. Slavakis is with the Department of Telecommunications Science and Technology, University of Peloponnese, Tripolis 22100, Greece (e-mail: slavakis@uop.gr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

with the presence of nonlinearities in the problem, e.g., [14].

The main contributions of the current paper can be summarized as follows:

- A new combine-project-adapt protocol is adopted. This is a modification of the combine-adapt protocol [9]. In each node, after the fusion and prior to the adaptation, a projection step is employed, whose purpose is to "harmonize" the received estimates from the neighborhood nodes. Its effect, as we will see, is to speed up convergence, and it fits naturally within the philosophy of projections that embraces the algorithmic family within which our solution is developed.
- The case of having some of the nodes malfunctioning is considered.
- Full monotone convergence to a single point, which satisfies the consensus requirement, is given in the Appendices.

The paper is organized as follows. In Section II, we describe the notation that will be used throughout the paper, as well as the general formulation of the problem, and in Section III the diffusion LMS is provided, whose simple structure can serve as a comparative standard, and the notion of consensus is discussed. In Section IV, the adaptive projected subgradient method, i.e., the kickoff point of our methodology, is presented, in simple geometric terms, and in the following section a novel diffusion algorithm is developed. In Section VI, we treat the case where a subset of the nodes set is malfunctioning. This problem is successfully attacked by adopting the Huber loss function [15]. Finally, in Section VII, a number of simulations are presented. In Appendix A, some basic mathematical preliminaries, concerning convex optimization, are briefly described and in Appendix B full proofs of our theorems are provided.

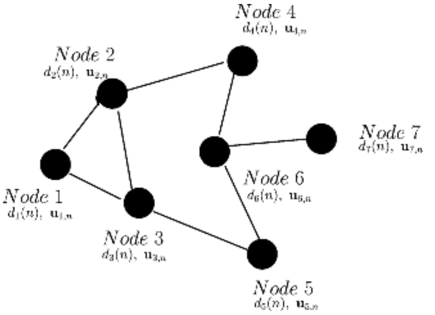
Finally, we provide with some notation which will be useful in the sequel. The set of real numbers and the set of non-negative integers are denoted by $\mathbb{R}$ and $\mathbb{N}$, respectively. Furthermore, we denote vectors by boldface letters, e.g., $\mathbf{u}$, and matrices with upper-case letters, whereas $\|\cdot\|$ stands for the Euclidean norm, in the vector case, and the 2-norm, in the matrix case. The notation $(\cdot)^T$ stands for the transposition operator, with $\mathrm{col}\{\ldots\}$ we denote the supervector which is formed by the stacking of the specified vectors, and $\mathrm{diag}\{\ldots\}$ denotes the block diagonal matrix, which consists of the matrices shown inside the brackets. Finally, we denote the Kronecker product of two matrices by $\otimes$.

II. Problem Formulation

Consider a fully distributed network of nodes, and our goal is to estimate an unknown vector $\mathbf{w}^* \in \mathbb{R}^m$ using measurements that are collected at each node. The node set is denoted by $\mathcal{N} := \{1, 2, \ldots, N\}$ and each node, k, at time n, has access to the measurements $d_k(n) \in \mathbb{R}$, $\mathbf{u}_{k,n} \in \mathbb{R}^m$. We assume that there exists a linear system, which generates these measurements according to the model

$$d_k(n) = \mathbf{u}_{k,n}^T \mathbf{w}^* + v_k(n) \tag{1}$$

where $v_k(n)$ is an additive noise process of zero mean and variance σ_k^2. A linear system, as defined by (1), is very common in adaptive filter theory, e.g., [16], [17]. Furthermore, we assume that every node is able to communicate with a subset of $\mathcal{N}$, say $\mathcal{N}_k$, which is known as the neighborhood of k. Throughout this paper, we assume that every node is a neighbor of itself, i.e., $k \in \mathcal{N}_k, \forall k \in \mathcal{N}$. This scenario is depicted in Fig. 1.



$Node\ 4$
$d_4(n),\ \mathbf{u}_{4,n}$
$Node\ 2$
$d_2(n),\ \mathbf{u}_{2,n}$
$Node\ 7$
$d_7(n),\ \mathbf{u}_{7,n}$
$Node\ 6$
$d_6(n),\ \mathbf{u}_{6,n}$
$Node\ 1$
$d_1(n),\ \mathbf{u}_{1,n}$
$Node\ 3$
$d_3(n),\ \mathbf{u}_{3,n}$
$Node\ 5$
$d_5(n),\ \mathbf{u}_{5,n}$

Fig. 1. An example of a diffusion network with $N = 7$ nodes.

The nodes cooperate with each other, which implies that the estimate obtained at a certain node will be exploited by its neighborhood; it turns out that such a scenario accelerates convergence and it also leads to a better steady state performance, compared to the case where every node works individually, e.g., [5], [9], [18]. Moreover, such a cooperation can provide asymptotic *consensus* ([18], [19]), i.e., all nodes will converge to the same estimate. In the literature, three cooperation schemes have been proposed:

- combine-adapt, e.g., [9], [20];
- adapt-combine, e.g., [19], [21], [22];
- consensus based, e.g., [8], [18].

In the combination stage, the information fusion takes place, as stated before, and the adaptation stage computes an estimate of the unknown parameter vector. In the consensus-based algorithms, there is no clear distinction between the combine and adapt steps.

The network's topology can be represented by the adjacency matrix $\mathcal{E}(n)$ $(N \times N)$, with elements

$$\mathcal{E}_{k,l}(n) = \begin{cases} 1, & \text{if } l \in \mathcal{N}_k \\ 0, & \text{if } l \notin \mathcal{N}_k. \end{cases}$$

It must be pointed out that the adjacency matrix, henceforth the network's topology, can be time varying, e.g., [19]. Additionally, we define the connectivity graph of the network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, which is assumed to be strongly connected, i.e., there is a path connecting any two nodes in it. This is a very common assumption that is adopted in distributed learning in diffusion networks, e.g., [8], [19], [23] and it is essential in order to reach consensus. Exploiting this topology, we can define the cooperation strategy through the combination matrix $\mathbf{C}(n)$, which is defined as

$$\mathbf{C}_{k,l}(n) = \begin{cases} c_{k,l}(n), & \text{if } l \in \mathcal{N}_k \\ 0, & \text{if } l \notin \mathcal{N}_k \end{cases}$$

where $c_{k,l}(n) \geq 0$ are known as the combination coefficients. A very important property of this matrix is: $\sum_{l \in \mathcal{N}_k} c_{k,l}(n) = 1$, $k = 1, \ldots, N$. Two typical examples of the combination matrix, $\mathbf{C}(n)$, are the Metropolis matrix, and the Nearest Neighbor matrix. The Metropolis matrix is defined as

$$c_{k,l}(n) = \begin{cases} \frac{1}{\max(|\mathcal{N}_k|, |\mathcal{N}_l|)}, & \text{if } l \in \mathcal{N}_k \text{ and } k \neq l \\ 1 - \sum_{l \in \mathcal{N}_k \setminus k} c_{k,l}, & \text{if } k = l \\ 0, & \text{otherwise} \end{cases}$$

whereas the Nearest Neighbor one as

$$c_{k,l}(n) = \begin{cases} \frac{1}{|\mathcal{N}_k|}, & \text{if } l \in \mathcal{N}_k \\ 0, & \text{otherwise} \end{cases}$$

where $|\mathcal{N}_k|$ stands for the number of neighbors of node k. Finally, if $\mathbf{C}(n)$ is properly chosen, as for example in the previous examples, then $\|\mathbf{C}(n)\| = 1$, a property that turns out to be very useful, as we will see later on (see also [9], [19]).

III. Diffusion LMS and the Consensus Matrix

A first approach to address the problem is the Diffusion LMS proposed in [9]. We present here the algorithm for the sake of comparison and also in order to serve as an introduction to the family of adaptive filters in distributed networks, since the LMS has established itself as the "standard", mainly due to its simplicity. The LMS is an algorithm that approximately minimizes recursively the mean square error

$$\min_{\mathbf{w}} E\left\{\left\|\mathbf{d} - \mathbf{U}^{\mathbf{T}}\mathbf{w}\right\|^2\right\}$$

where $E\{\cdot\}$ denotes the expectation operator, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$ is a matrix of dimension $m \times N$ having as columns the input vectors, $\mathbf{u}_k$, where the dependence on n has been dropped, and $\mathbf{d} = [d_1, d_2, \ldots, d_N]^T \in \mathbb{R}^N$. The combine-adapt diffusion LMS, for every node, is given by the following:

$$\boldsymbol{\phi}_k(n) = \sum_{l \in \mathcal{N}_k} c_{k,l}(n+1)\mathbf{w}_l(n) \qquad (2)$$

$$\mathbf{w}_k(n+1) = \boldsymbol{\phi}_k(n) + \mu_k(n+1) \times (d_k(n+1) - \mathbf{u}_{k,n+1}^T\boldsymbol{\phi}_k(n))\mathbf{u}_{k,n+1} \qquad (3)$$

where $\mu_k(n+1)$ stands for the local step-size. It is readily seen that (2) constitutes the fusion step (combine) and (3) the adaptation step (adapt). One can rewrite the previous equations for the whole network

$$\begin{cases} \boldsymbol{\phi}(n) = \mathbf{G}(n+1)\mathbf{w}(n), \\ \mathbf{w}(n+1) = \boldsymbol{\phi}(n) + \mathbf{M}_{n+1}\mathbf{U}_{n+1}(\mathbf{d}(n+1) - \mathbf{U}_{n+1}^T\boldsymbol{\phi}(n)) \end{cases}$$

where $\mathbf{w}(n) = \mathrm{col}\{\mathbf{w}_1(n), \ldots, \mathbf{w}_N(n)\} \in \mathbb{R}^{Nm\times 1}$, $\boldsymbol{\phi}(n) = \mathrm{col}\{\boldsymbol{\phi}_1(n), \ldots, \boldsymbol{\phi}_N(n)\} \in \mathbb{R}^{Nm\times 1}$, $\mathbf{U}_{n+1} = \mathrm{diag}\{\mathbf{u}_{1,n+1}, \ldots, \mathbf{u}_{N,n+1}\}\,(Nm\times N)$, $\mathbf{M}_{n+1} = \mathrm{diag}\{\mu_1(n+1)\mathbf{I}_m, \ldots, \mu_N(n+1)\mathbf{I}_m\}\,(Nm \times Nm)$, $\mathbf{G}(n+1) = \mathbf{C}(n+1) \otimes \mathbf{I}_m\,(Nm \times Nm)$ and $\mathbf{I}_m$ is the identity matrix of size $m \times m$. From now on, we will refer to $\mathbf{G}(n)$ as the *Consensus Matrix* . Some properties of this matrix are [19]:

1) $\|\mathbf{G}(n)\| = 1$.
2) Any consensus matrix can be decomposed as

$$\mathbf{G}(n) = \mathbf{X}(n) + \mathbf{B}\mathbf{B}^T, \qquad (4)$$

where $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m]$ is an $Nm \times m$ matrix, and $\mathbf{b}_k = \frac{(\mathbf{1}_N \otimes \mathbf{e}_k)}{\sqrt{N}}$, $\mathbf{e}_k$ is a $m \times 1$ vector of zeros except the kth entry, which is one and $\mathbf{X}(n)$ is an $Nm \times Nm$ matrix such that $\|\mathbf{X}(n)\| < 1$.
3) $\mathbf{G}(n)\mathbf{x} = \mathbf{x}, \forall \mathbf{x} \in \mathcal{O} := \{\mathbf{z} \in \mathbb{R}^{Nm} : \mathbf{z} = \mathrm{col}\{\mathbf{p}, \ldots, \mathbf{p}\},\ \mathbf{p} \in \mathbb{R}^m\}$. The latter set is the so called consensus subspace of dimension m, where $\mathbf{b}_k,\ k = 1, \ldots, m$, constitute a basis for this set. From the last argument, it can be readily verified that the orthogonal projection of a vector, $\mathbf{x}$, onto this linear subspace is given by $P_{\mathcal{O}}(\mathbf{x}) := \mathbf{B}\mathbf{B}^T\mathbf{x},\ \forall \mathbf{x} \in \mathbb{R}^{Nm}$.

IV. Adaptive Projection Algorithm Using Projections Onto Hyperslabs

The algorithms to be developed belong to the family of the adaptive projected subgradient method (APSM) presented in [11] and generalized in [13]. At the heart of the method lies the following concept. Instead of a single loss function, that is minimized over the whole set of measurements, each time instant is treated separately. Given the measurement pair $(\mathbf{u}_{n+1}, d(n+1)) \in \mathbb{R}^m \times \mathbb{R}$, at time $n+1$, the designer quantifies his/her perception of loss, with respect to the received measurement pair, by a "local" loss function, which can also be considered to be time varying (e.g., [19]). This "local" loss defines a region (set of points), which is also known as a *property set* , where the estimate of the unknown vector would be desirable to lie, in order to be considered in *agreement* with the current measurements (i.e., low error, smaller than a predefined threshold). Assuming the "local" loss function $\Theta_{n+1} : \mathbb{R}^m \rightarrow \mathbb{R} : \mathbf{w} \mapsto \Theta_{n+1}(\mathbf{w})$ to be convex, the respective property set, is nothing but the associated 0-th level set defined as $\mathrm{lev}_{\leq 0}\Theta_{n+1} = \{\mathbf{w} \in \mathbb{R}^m : \Theta_{n+1}(\mathbf{w}) \leq 0\}$, which is also convex. The goal is to find a point in the intersection of all these property sets, one per time instant. This is the same goal that defines the classical projections onto convex sets (POCS) theory. However, here, the number of the involved convex sets is infinite. So, from this point of view, this theory can be considered as a generalization of the POCS. The key algorithmic recursion is

$$\mathbf{w}(n+1) = \begin{cases} \mathbf{w}(n) - \lambda(n+1)\frac{\Theta_{n+1}(\mathbf{w}(n))}{\|\Theta'_{n+1}(\mathbf{w}(n))\|^2}\Theta'_{n+1}(\mathbf{w}(n)), \\ \qquad\qquad \text{if } \Theta'_{n+1}(\mathbf{w}(n)) \neq \mathbf{0} \\ \mathbf{w}(n), \qquad\quad \text{if } \Theta'_{n+1}(\mathbf{w}(n)) = \mathbf{0} \end{cases} \qquad (5)$$

where $\Theta'_{n+1}(\mathbf{w}(n))$ is the subgradient of Θ_{n+1} at the current estimate $\mathbf{w}(n)$ (the definition as well as the geometrical interpretation of the subgradient are given in Appendix A) and $\lambda(n+1) \in (0, 2)$. The goal is to push the available estimate towards the current property set, which is defined by the measurement pair at time $n+1$. This is illustrated in Fig. 2(b), where being at $\mathbf{w}$, a support hyperplane defined by a subgradient (the gradient) divides $\mathbb{R}^m$ in two halfspaces (see Appendix A). Projecting $\mathbf{w}$ onto the halfspace, where the level set lies, guarantees that we get closer to the level set, where a solution lies.

Before we proceed, it is interesting and it will help the reader to grasp the reasoning behind the algorithms to be developed, to notice a difference between Fig. 2(a) and (b). In Fig. 2(a), the 0-th level set is a hyperslab (see Appendix A), and one can reach it in a single step, via a single projection of $\mathbf{w}$ onto the hyperslab. In other words, APSM algorithm in this case, breaks down to a simple projection onto the hyperslab. The case of Fig. 2(b) is different, where in order to approach the level set, APSM results in a succession of projections onto a sequence of halfspaces. All these simple intuitive geometric concepts will be theoretically documented in the Appendix B.

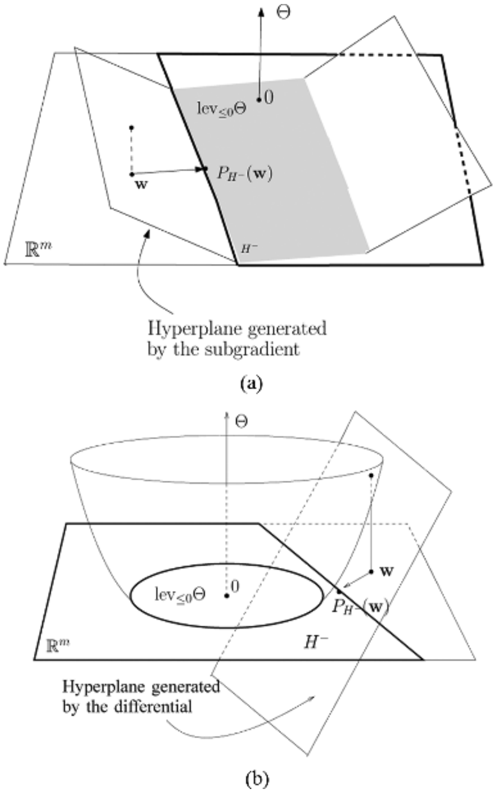


Fig. 2. (a) A cost function whose 0-th level set is a hyperslab (b) A cost function and the supporting hyperplane generated by the differential.

Remark 1: Note that the APSM stems for Polyak's algorithm [24]. Nevertheless, in contrast to Polyak's algorithm, where the cost function to be minimized is fixed, here it may be time varying, a fact that allows the adopted algorithmic scheme to be applicable in dynamic and time-adaptive scenarios. Furthermore, despite the fact that the geometrical properties remind us of the Newton-Raphson algorithm, the two schemes are different. This holds, since in the Newton-Raphson algorithm a matrix inversion is needed in order to update the recursion, which is not the case in the APSM. Moreover, as stated before, in the APSM based algorithms differentiability of the cost function is not essential. □

As it has already been stated, if the property set is a hyperslab, then the recursion breaks down to a projection on this hyperslab. This is the case to be considered in this section. Given the measurement pair $(\mathbf{u}_{n+1}, d(n+1)) \in \mathbb{R}^m \times \mathbb{R}$, the associated property set is defined as

$$S_{n+1} = \{\mathbf{w} \in \mathbb{R}^m : |d(n+1) - \mathbf{w}^T\mathbf{u}_{n+1}| \leq \epsilon\} \tag{6}$$

where ϵ is a user-defined parameter and it is chosen such that to account for the noise and it will be discussed in the next section. The previous hyperslab is the 0-th level set of the loss function

$$\Theta_{n+1}(\mathbf{w}) = \max\{0, |d(n+1) - \mathbf{w}^T\mathbf{u}_{n+1}| - \epsilon\},\ \mathbf{w} \in \mathbb{R}^m. \tag{7}$$

Every $\mathbf{w}$ that lies in the hyperslab scores a zero loss and it will be considered to be in *agreement* with the current set of measurements. Such cost functions are met in robust statistics and have been popularized in the context of SVM regression, e.g., [25].

The algorithm whose goal is to push the currently available estimate towards the hyperslab, which is defined by the current set of measurement points, is given by

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n+1)\left(P_{S_{n+1}}(\mathbf{w}(n)) - \mathbf{w}(n)\right) \tag{8}$$

where $P_{S_{n+1}}$ is the projection operator associated with the hyperslab (6) and it is provided in Appendix A and $\mu(n+1) \in (0,2)$. It is a matter of simple algebra to see that (8) holds if (7) is used as a loss function in (5). Moreover, as it is always the case with adaptive algorithms, recursion steps coincide with time steps. This algorithm can be further generalized if one projects onto the q more recent hyperslabs and then take a convex combination of the result. The effect of this is to speed up convergence, and reminds us of the rationale behind the APA algorithm, e.g., [16]. The resulting algorithm then becomes

$$\mathbf{w}(n+1) = \mathbf{w}(n) \\ +\mu(n+1)\left(\sum_{j \in \mathcal{J}_{n+1}} \omega_j P_{S_j}(\mathbf{w}(n)) - \mathbf{w}(n)\right) \tag{9}$$

where $\mathcal{J}_{n+1} := \overline{\max\{0, n-q+2\}, n+1}$ and the overline symbol, for given integers j_1, j_2 with $j_1 \leq j_2$, is defined as $\overline{j_1, j_2} := \{j_1, j_1+1, \ldots, j_2\}$. Moreover, $\sum_{j \in \mathcal{J}_{n+1}} \omega_j = 1$ and $\mu(n+1)$ is chosen so as to guarantee convergence. As it has been shown, the scheme converges to the intersection of all the hyperslabs (hence it is in agreement with all the measurements) with the possible exception of a finite number of them, which allows for the presence of outliers [11]–[13]. In the next section, we present a diffusion version of it.

V. Diffusion Adaptive Algorithm Using Projection Onto Hyperslabs

Consider the problem described in Section II. The goal is to derive (9) for every node of the network, following the rationale behind the diffusion LMS, as presented in Section III. Although this is a possibility, here we have added an extra step, that follows the combination stage and precedes the adaptation one. As it will become apparent in the simulations section, such a step turns out to be beneficial to the convergence performance, at the expense of the minimal cost of an extra projection onto a hyperslab $S'_{k,n+1}$, which is defined as

$$S'_{k,n+1} = \{\mathbf{w} \in \mathbb{R}^m : |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \epsilon'_k\}$$

where $\epsilon'_k > \epsilon_k$ and ϵ_k is the user defined parameter associated with the hyperslabs, that will be used in the adaptation step at node k. The algorithm comprises the following steps:

1) The estimates from the nodes that belong to $\mathcal{N}_k$ are received and convexly combined.
2) The aggregate is first projected onto the hyperslab $S'_{k,n+1}$.
3) The adaptation step is performed.

Algorithm 1:

$$\boldsymbol{\phi}_k(n) = \sum_{l \in \mathcal{N}_k} c_{k,l}(n+1)\mathbf{w}_l(n) \tag{10}$$

$$\mathbf{z}_k(n) = P_{S'_{k,n+1}}\left(\boldsymbol{\phi}_k(n)\right) \tag{11}$$

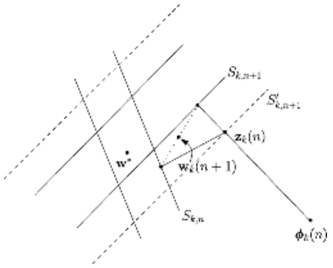


Fig. 3. Illustration of an iteration for the case of $q = 2$. The aggregate is projected onto an external hyperslab and the result, $\mathbf{z}_k(n)$, is used in the adaptation step. Note that $\mathbf{z}_k(n)$ is projected onto $S_{k,n+1}$ and $S_{k,n}$, and the projections are combined together. The update estimate, $\mathbf{w}_k(n+1)$ lies closer to the intersection of the hyperslabs, compared to $\boldsymbol{\phi}_k(n)$. Note, also, that hyperslab $S'_{k,n+1}$ contains $S_{k,n+1}$.

$$\mathbf{w}_k(n+1) = \mathbf{z}_k(n) + \mu_k(n+1) \times \left(\sum_{j \in \mathcal{J}_{n+1}} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_k(n)) - \mathbf{z}_k(n) \right). \tag{12}$$

The geometry for the case of $q = 2$ is shown in Fig. 3. The aggregate $\boldsymbol{\phi}_k(n)$ is first projected onto $S'_{k,n+1}$ to provide $\mathbf{z}_k(n)$. The latter is the point that gets involved in the adaptation step, which consists of the projections onto $S_{k,n+1}$ and $S_{k,n}$, and in their subsequent convex combination in accordance to the APSM rationale. As it will be shown in the Appendix B, convergence is guaranteed if $\mu_k(n+1) \in (0, 2\mathcal{M}_k(n+1))$ where

$$\mathcal{M}_k(n+1) = \begin{cases} \dfrac{\sum_{j \in \mathcal{J}_{n+1}} \omega_{k,j} \left\| P_{S_{k,j}}(\mathbf{z}_k(n)) - \mathbf{z}_k(n) \right\|^2}{\left\| \sum_{j \in \mathcal{J}_{n+1}} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_k(n)) - \mathbf{z}_k(n) \right\|^2}, & \\ \qquad \text{if } \sum_{j \in \mathcal{J}_{n+1}} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_k(n)) \neq \mathbf{z}_k(n) & \\ 1, \qquad \text{otherwise.} & \end{cases} \tag{13}$$

Also, $\mathcal{M}_k(n+1) \geq 1$ [26], hence $\mu_k(n+1) = 1$ is an acceptable step-size $\forall n \in \mathbb{N}$.

Before we proceed further, let us "translate" (10)–(12) from the local to a global form to include all nodes. It is a matter of simple algebra to see that

$$\mathbf{z}(n) = \begin{bmatrix} \mathbf{z}_1(n) \\ \vdots \\ \mathbf{z}_N(n) \end{bmatrix} = \begin{bmatrix} P_{S'_{1,n+1}} \left(\sum_{l \in \mathcal{N}_1} c_{1,l}(n+1) \mathbf{w}_l(n) \right) \\ P_{S'_{2,n+1}} \left(\sum_{l \in \mathcal{N}_2} c_{2,l}(n+1) \mathbf{w}_l(n) \right) \\ \vdots \\ P_{S'_{N,n+1}} \left(\sum_{l \in \mathcal{N}_N} c_{N,l}(n+1) \mathbf{w}_l(n) \right) \end{bmatrix} \tag{14}$$

$$\mathbf{w}(n+1) = \mathbf{z}(n) + \mathbf{M}_{n+1} \times \begin{bmatrix} \sum_{j \in \mathcal{J}_{n+1}} \omega_{1,j} P_{S_{1,j}}(\mathbf{z}_1(n)) - \mathbf{z}_1(n) \\ \sum_{j \in \mathcal{J}_{n+1}} \omega_{2,j} P_{S_{2,j}}(\mathbf{z}_2(n)) - \mathbf{z}_2(n) \\ \vdots \\ \sum_{j \in \mathcal{J}_{n+1}} \omega_{N,j} P_{S_{N,j}}(\mathbf{z}_N(n)) - \mathbf{z}_N(n) \end{bmatrix} \tag{15}$$

where $\mathbf{M}_{n+1} = \operatorname{diag}\{\mu_1(n+1)\mathbf{I}_m, \mu_2(n+1)\mathbf{I}_m, \ldots, \mu_N(n+1)\mathbf{I}_m\}$.

Observe that the resulting scheme is structurally simple. It consists of two vector equations, one for the combination/fusion and one for the update. The main operations that are involved are projections. The complexity per time update amounts to $O(qm)$ in every node. Moreover, in a parallel processing environment, the q projections can take place concurrently.

As it will be established in Appendix B, the algorithm (14) and (15) enjoys a number of nice convergence properties such as monotonicity, strong convergence to a point and consensus. To prove these properties the following assumptions must hold.

Assumptions:

a) There exists a non-negative integer, say n_0, for which $\Omega = \bigcap_{n \geq n_0} \Omega(n) \neq \emptyset$, where $\Omega(n) = \bigcap_{k \in \mathcal{N}} \bigcap_{j \in \mathcal{J}_{n+1}} S_{k,j}$. This means that the hyperslabs share a non empty intersection. However, it is possible for a finite number of them, n_0, not to share a common intersection. This is important, since the presence of a finite number of outliers does not affect convergence. A case where such an assumption holds true is whenever the noise is bounded, and the width of the hyperslabs, determined by the parameter ϵ_k, is chosen appropriately so as to contain $\mathbf{w}^*$.

b) The local stepsize $\mu_k(n) \in (0, 2\mathcal{M}_k(n))$, $k = 1, \ldots, N$. As it is always the case with adaptive algorithms, the adaptation step must lie within an interval, in order to guarantee convergence. Here, this interval is computed by the algorithm itself.

c) Let $\varepsilon_1 > 0$ be a sufficiently small constant such that $\mu_k(n) \in [\varepsilon_1 \mathcal{M}_k(n), \mathcal{M}_k(n)(2 - \varepsilon_1)]$, $k = 1, \ldots, N$.

d) In order to guarantee consensus the following statement must hold: $\epsilon'_k > \epsilon_k$, $\forall k \in \mathcal{N}$.

e) Let us define $\mathfrak{C} := \mathcal{O} \cap \boldsymbol{\Omega}$, where the Cartesian product space, $\boldsymbol{\Omega}$, is defined as $\boldsymbol{\Omega} = \underbrace{\Omega \times \cdots \times \Omega}_{N}$ and $\mathcal{O}$ has been defined in property 3) in Section III. We assume that $\operatorname{ri}_{\mathcal{O}}(\mathfrak{C}) \neq \emptyset$, where $\operatorname{ri}_{\mathcal{O}}(\mathfrak{C})$ stands for the relative interior of $\mathfrak{C}$ with respect to $\mathcal{O}$, and its definition is given in Appendix A.

Theorem 1: For any $\underline{\hat{\mathbf{w}}} \in \mathfrak{C}$, which is of the form $\underline{\hat{\mathbf{w}}} = \left[\hat{\mathbf{w}}^T, \ldots, \hat{\mathbf{w}}^T\right]^T \in \mathbb{R}^{Nm}$, with $\hat{\mathbf{w}} \in \Omega$, the following hold true.

1) **Monotone Approximation**. Under assumptions a) and b)

$$\|\mathbf{w}(n+1) - \underline{\hat{\mathbf{w}}}\| \leq \|\mathbf{w}(n) - \underline{\hat{\mathbf{w}}}\|, \forall n \geq n_0. \tag{16}$$

The previous inequality yields that, the distance of $\mathbf{w}(n)$ from any point $\underline{\hat{\mathbf{w}}} \in \mathfrak{C}$ (that comprises our solution space) is a nonincreasing function of the time adaptation step, n.

2) **Asymptotic Minimization**. Monotone approximation "informs" us, as stated before, that every iteration step takes the current update closer to a desired solution. However, this cannot guarantee that asympotically the algorithm converges to a point that lies close to the intersection Ω. If assumptions (1), (3) hold true then

$$\lim_{n \longrightarrow \infty} d(\mathbf{w}_k(n), \Omega(n)) = 0, \ \forall k \in \mathcal{N}$$

where $d(\mathbf{w}_k(n), \Omega(n))$ is the distance of $\mathbf{w}_k(n)$ from $\Omega(n)$ (the definition of the distance function can be found in

Appendix A). In other words, the distance of the obtained estimates, in each one of the nodes, from the intersection of the respective hyperslabs that define the solution set, tends asymptotically to zero.

3) **Asymptotic Consensus**. It has been shown, [19], that in order to achieve asymptotic consensus, i.e., [27]

$$\lim_{n \longrightarrow \infty} \|\mathbf{w}_k(n) - \mathbf{w}_l(n)\| = 0, \ \forall k, l \in \mathcal{N} \tag{17}$$

the following must hold true:

$$\lim_{n \longrightarrow \infty} [(\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T)\mathbf{w}(n)] = \mathbf{0},$$

where $\mathbf{I}_{Nm}$ stands for the $Nm \times Nm$ identity matrix. The previous statement is true under assumptions a), c), and d).

4) **Strong Convergence**. If a), c), d), and e) hold true, then there exists $\hat{\mathbf{w}}_* \in \mathcal{O}$ such that

$$\lim_{n \longrightarrow \infty} \mathbf{w}(n) = \hat{\mathbf{w}}_*.$$

In other words, the sequence generated by (15) converges strongly to a point in $\mathcal{O}$.

Proof: The proof is given in Appendix B. ■

Remark 2: A projection based algorithm of the adapt-combine type has been developed in [19]. In contrast, our algorithm follows the combine-project-adapt philosophy. This scenario gives us the advantage of being able to accommodate the extra projection. The notion behind this extra step is to "harmonize", at each node, the information that is sensed locally with the information transmitted by the neighboring nodes. Although all nodes search for the same unknown vector (i.e., $\mathbf{w}^*$), the statistics of the regressors are, in general, different. For this reason, by projecting the aggregate (which is the information collected from the neighborhood) onto a hyperslab, i.e., $S'_{k,n}$, which is constructed using information that is sensed locally, we push the aggregate closer to the feasible region and the convergence is accelerated, which is verified by the experiments. Note that if we let $P_{S'_{k,n+1}}$ be the identity mapping, then the algorithm conforms to the simple combine-adapt cooperation protocol. Another notable difference with [19] is that the theoretical analysis of the algorithm is different and we have proved strong convergence of our algorithm, which was not the case with [19]. □

The choice of the value of the user-defined parameter, ϵ_k, is dictated by the noise variance. If its value is very small, the width of the involved hyperslabs is small and the convergence is slow. If the value is large, the convergence speeds up, but the final error floor increases. This tradeoff follows the same trend that underlies the choice of parameters in most adaptive learning schemes.

VI. Introducing Robustness to Cope With a Failure of Nodes

Consider a scenario, in which some of the nodes are damaged and the associated observations are very noisy[1]. In such cases, the use of loss functions suggested in the framework of robust-statistics are more appropriate to cope with outliers. A popular cost function of this family is the Huber cost function, e.g., [15], [28], defined as,

$$\tilde{\Theta}_{k,n+1}(\mathbf{w}) = \begin{cases} 0, \quad \text{if } |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \tilde{\gamma}_k \\ \frac{1}{2}|d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}|^2 - \frac{\tilde{\gamma}_k^2}{2}, \\ \qquad \text{if } \tilde{\gamma}_k < |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \tilde{c}_k \\ \tilde{c}_k|d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| - \frac{\tilde{\gamma}_k^2}{2} - \frac{\tilde{c}_k^2}{2} \\ \qquad \text{otherwise.} \end{cases} \tag{18}$$

The one-dimensional version of it is illustrated in Fig. 4. The use of this function in the context of sensor networks has also been suggested in [15]. Let us take a closer look at (18). First of all, whenever $|d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \tilde{\gamma}_k$, we assume that our cost function scores a zero penalty and the non-negative, user-defined parameter, $\tilde{\gamma}_k$, defines the 0-th level set (property set) of the function. On the contrary, if $\tilde{\gamma}_k < |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \tilde{c}_k$, where $\tilde{c}_k > \tilde{\gamma}_k$ is also a user defined parameter, then the estimate scores a non-zero penalty, with a square dependence on the error. Finally, in the case when $|d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| > \tilde{c}_k$, the measurements have probably occurred from a corrupted node and the cost function now changes to a linear dependence so as to treat it as an outlier.

In order to derive the new algorithm around the cost in (18), all that has to change, with respect to the algorithm which was previously developed, are the projection operators. Instead of projecting onto hyperslabs, one has to project onto halfspaces, denoted as $H^-_{k,n+1}$, which are formed by the intersections of the supporting hyperplanes (associated with the subgradients) and the space where the solution lies, as already discussed in Section IV and illustrated in Fig. 2(b). Algebraically, this is done by the APSM formula (5), i.e.

$$P_{H^-_{k,n+1}}(\mathbf{w}) = \begin{cases} \mathbf{w} - \frac{\tilde{\Theta}_{k,n+1}(\mathbf{w})}{\|\tilde{\Theta}'_{k,n+1}(\mathbf{w})\|^2}\tilde{\Theta}'_{k,n+1}(\mathbf{w}), \\ \qquad \text{if } \tilde{\Theta}'_{k,n+1}(\mathbf{w}) \neq 0 \\ \mathbf{w}, \qquad \text{otherwise.} \end{cases} \tag{19}$$

The subgradient $\tilde{\Theta}'_{k,n+1}$ of the loss function is given by [29] shown in (20) at the bottom of the next page where $\mathrm{sgn}(\cdot)$ denotes the sign function.

We can also include the extra projection step, described in the previous section, by introducing a modified version of (18) and following a similar rationale as in Section V. However, instead of projecting the aggregate $\boldsymbol{\phi}_k(n)$ onto an external hyberslab, we project it onto a halfspace that is generated by a properly modified cost function (Fig. 4). To be more, specific we define

$$\hat{\Theta}_{k,n+1}(\mathbf{w}) = \begin{cases} 0, \quad \text{if } |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \hat{\gamma}_k \\ \frac{1}{2}|d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}|^2 - \frac{\hat{\gamma}_k^2}{2} \\ \qquad \text{if } \hat{\gamma}_k < |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \hat{c}_k \\ \hat{c}_k|d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| - \frac{\hat{\gamma}_k^2}{2} - \frac{\hat{c}_k^2}{2} \\ \qquad \text{otherwise} \end{cases}$$

where $\hat{c}_k > \hat{\gamma}_k$ and $\hat{\gamma}_k > \tilde{\gamma}_k$. The latter condition is required to guarantee consensus. The projection of an arbitrary point onto the halfspace $H'^-_{k,n+1}$ is similar to (19) and the algorithm for the whole network is similar to the one given in (14), (15) with the

[1] We assume that the noise remains additive and white. However, its standard deviation becomes larger.

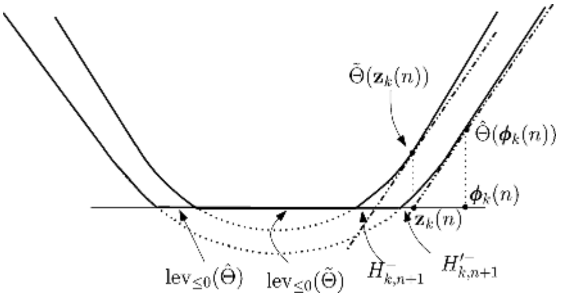

$\tilde{\Theta}(\mathbf{z}_k(n))$
$\hat{\Theta}(\boldsymbol{\phi}_k(n))$
$\boldsymbol{\phi}_k(n)$
$\mathbf{z}_k(n)$
$\mathrm{lev}_{\leq 0}(\hat{\Theta})$
$\mathrm{lev}_{\leq 0}(\tilde{\Theta})$
$H^-_{k,n-1}$
$H'^-_{k,n+1}$

Fig. 4. Illustration of the cost functions $\tilde{\Theta}(\cdot),\ \hat{\Theta}(\cdot)$. The aggregate, $\boldsymbol{\phi}_k(n)$, is projected onto $H'^-_{k,n+1}$, which is the intersection of the hyperplane associated with the subgradient at $\hat{\Theta}(\boldsymbol{\phi}_k(n))$ with the space $\mathbb{R}^m$, to provide $\mathbf{z}_k(n)$. In the sequel, $\mathbf{z}_k(n)$ is used into the adaptation step.

slight difference that the involved hyperslabs have been replaced by halfspaces.

As in the case of the hyperslab projection algorithm, the algorithmic scheme, which employs the Huber loss function, enjoys monotonicity, asymptotic optimality, asymptotic consensus and strong convergence. Obviously, the assumptions under which the algorithm enjoys these convergence properties have to change compared to the algorithm of the previous section. More specifically, Ω now becomes $\Omega' = \bigcap_{n\geq n_0} \Omega'(n) \neq \emptyset$ where $\Omega'(n) = \bigcap_{k\in\mathcal{N}} \bigcap_{j\in\mathcal{J}_{n+1}} H^-_{k,j}$. Furthermore, the stepsizes must lie inside the interval with upper bound determined as in the previous algorithm, with the difference that $\mathcal{M}_k(n)$ is replaced by a parameter determined by the projection operators onto the respective halfspaces. Finally, the assumption regarding the consensus is now $\hat{\gamma}_k > \tilde{\gamma}_k$. Such a choice guarantees that $\mathrm{lev}_{\leq 0}\tilde{\Theta}_{k,n+1} \subset \mathrm{lev}_{\leq 0}\hat{\Theta}_{k,n+1}$, which is required in order to prove consensus (see Appendix B).

Remark 3: Note that although the loss function used in this section is quite different compared to the hyperslabs used before, both algorithms, i.e., the one built around the Huber loss function and the one given in (14) and (15), are of the same form. The only difference lies in the projection operators. Moreover, the theoretical analysis is exactly the same. All that matters is the property of convexity. This is a major advantage of the methodology behind this set theoretic approach.

VII. Numerical Examples

In this section, we present simulation results in order to study the comparative performance of the developed algorithms with respect to previously reported schemes. The general framework of our experiments is the system identification task in diffusion networks. A linear system described by (1) is adopted and our goal is to estimate the unknown vector $\mathbf{w}^*$ using the measurements $d_k(n)$, $\mathbf{u}_{k,n}$. The components of the regression vectors, i.e., $\mathbf{u}_{k,n} = [u_{k,n}, \ldots, u_{k,n-m+1}]$, are generated according to

$$u_{k,n} = \psi_k u_{k,n-1} + \chi_{k,n},\ \forall k \in \mathcal{N},$$

where $\psi_k \in (0, \psi_u)$ and it is distributed according to the uniform distribution and ψ_u is a parameter that we alter throughout our experiments in order to investigate the behaviour of the algorithms for cases where the regressors are strongly or weakly correlated. Finally, $\chi_{k,n}$ is Gaussian with unit variance. The standard deviation of the noise $v_k(n)$, which is assumed to be white and Gaussian, equals to $\sigma_k = b_k\sigma_u$ where $b_k \in (0,1)$ under the uniform distribution and σ_u is user-defined and will change throughout our experiments. In order to construct the network, the following strategy has been followed. A certain node, say k, is connected to any other node with probability equal to 0.3, and it is connected to itself with probability 1. Additionally, the combination matrix $\mathbf{C}(n)$ is constructed according to the Metropolis rule. Finally, the adopted performance metrics used are as follows:

- mean square error (MSE), which is defined as $\sum_{k=1}^{N} \frac{(d_k(n) - \mathbf{u}^T_{k,n}\mathbf{w}_k(n))^2}{N}$,
- mean square deviation (MSD), which is defined as $\sum_{k=1}^{N} \frac{\|\mathbf{w}^* - \mathbf{w}_k(n)\|^2}{N}$.

The experiments are averaged over 100 realizations for smoothing purposes.

We compare the proposed algorithm 1 with a) the adapt-combine LMS of [22] (A-C LMS), b) the consensus based LMS [18] (D-LMS), and c) with the Adaptive Projected Subgradient Method in Diffusion networks (APSMd) proposed in [19]. In the first experiment, we consider a diffusion network with N=20 nodes, and the unknown vector $\mathbf{w}^*$ is of dimension $m = 60$. The noise profile for this network is obtained with $\sigma_u = 10^{-3}, \psi_u = 0.5$. The parameter $\epsilon = \sqrt{2}\sigma_k$ for both the proposed algorithm 1 and for the APSMd and the parameter $\epsilon'_k = 2\epsilon_k$. Furthermore, the number of hyperslabs onto which we project, in each step, is $q = 8$, whereas the convex combination multipliers are all equal, i.e., $\omega_{k,n} = \frac{1}{q}$, and we let $\mu_k(n) = 1,\ k = 1, \ldots, N$, for algorithm 1 and APSMd. It has been experimentally verified that for such a choice, the projection based algorithms exhibit very good convergence performance as well as a low steady state error floor. For the A-C LMS, the stepsize equals to $\mu_{LMS} = 0.01$, whereas for D-LMS $\mu_{D-LMS} = 0.006$. The stepsizes in the LMS-based algorithms are chosen so that the algorithms reach the same steady state error floor in the MSE sense. Throughout our experiments, if we let $P_{S'_{k,n}}$ be the identity mapping, it turns out that the proposed algorithm 1 and the

$$\tilde{\Theta}'_{k,n+1}(\mathbf{w}) = \begin{cases} \mathbf{0}, & \text{if } |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| \leq \tilde{\gamma}_k \\ \kappa(d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1})(-\mathbf{u}_{k,n+1}) \text{ with}\kappa \in [0,1], & \text{if } |d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1}| = \tilde{\gamma}_k \\ (d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1})(-\mathbf{u}_{k,n+1}), & \text{if } \tilde{\gamma}_k < |d_k(n+1) - \mathbf{w}^T\ \mathbf{u}_{k,n+1}| \leq \tilde{c}_k \\ \tilde{c}_k \mathrm{sgn}(d_k(n+1) - \mathbf{w}^T\mathbf{u}_{k,n+1})(-\mathbf{u}_{k,n+1}), & \text{otherwise} \end{cases} \quad (20)$$

APSMd have similar performance. Hence, as it can be seen in Fig. 5(a) and (b), the extra projection onto the hyperslab $S'_{k,n}$, which adds an $O(m)$ expense on the computational complexity of the algorithm for each node, enhances the results compared to the other algorithms, as it accelerates the convergence speed for the same error floor. Moreover, the proposed algorithm 1 outperforms the LMS-based algorithms and the APSMd, as it achieves a better steady state error floor in the MSD sense. Nevertheless, compared to the LMS-based algorithms, the proposed algorithm 1 and the APSMd require knowledge on the noise statistics, i.e., σ_k, $\forall k \in \mathcal{N}$. Moreover, for $q > 1$, the projection-based algorithms require some extra memory in order to store past data. We would also like to mention, that through experiments, we observed small discrepancies between the steady state performances of the nodes, despite the fact that the noise statistics may be different. This is also known as equalization property and it is a common effect met in diffusion based algorithms, e.g., [9], [18].

In the second experiment, [see Fig. 6(a) and (b)], the parameters remain the same as in the previous one. However, we choose a larger ψ_u, namely 0.9, in order to compare the algorithms in a more correlated environment. Furthermore, we alter the step-sizes for the LMS-based algorithms. More specifically, the values $\mu_{LMS} = 0.007$ and $\mu_{D-LMS} = 0.002$ were chosen in a similar philosophy as in the previous experiment. As it is expected, the LMS-based algorithms result in worse performance compared to the previous example of less correlated signals, something that holds true also in the classical LMS case [16]. This performance trend can be seen both in the MSE and the MSD curves, as algorithm 1 outperforms significantly the LMS-based algorithms. Moreover, as it was the case in the first experiment, it can be seen that the extra projection step accelerates the convergence speed of proposed algorithm 1 compared to the APSMd.

The scenario in the third experiment [see Fig. 7(a) and (b)] is the same as the one in the first experiment, but now after a number of iterations there is a sudden change in the channel. At time instant $n = 1800$, $\mathbf{w}^*$ changes to $-\mathbf{w}^*$. This is a popular experiment in adaptive filter theory in order to test the tracking performance of the algorithm. As it is by now well established, fast convergence speed does not necessarily guarantee a good tracking performance [16]. It can be readily seen that the proposed algorithm 1 shows a large capacity for tracking ability, when a sudden change in the channel takes place. More specifically, until the time instant at which $\mathbf{w}^*$ changes, the performance of the algorithms coincides with that of Fig. 5(a) and (b). After the sudden change, the proposed algorithm 1 exhibits the best tracking performance to the common steady state error floor.

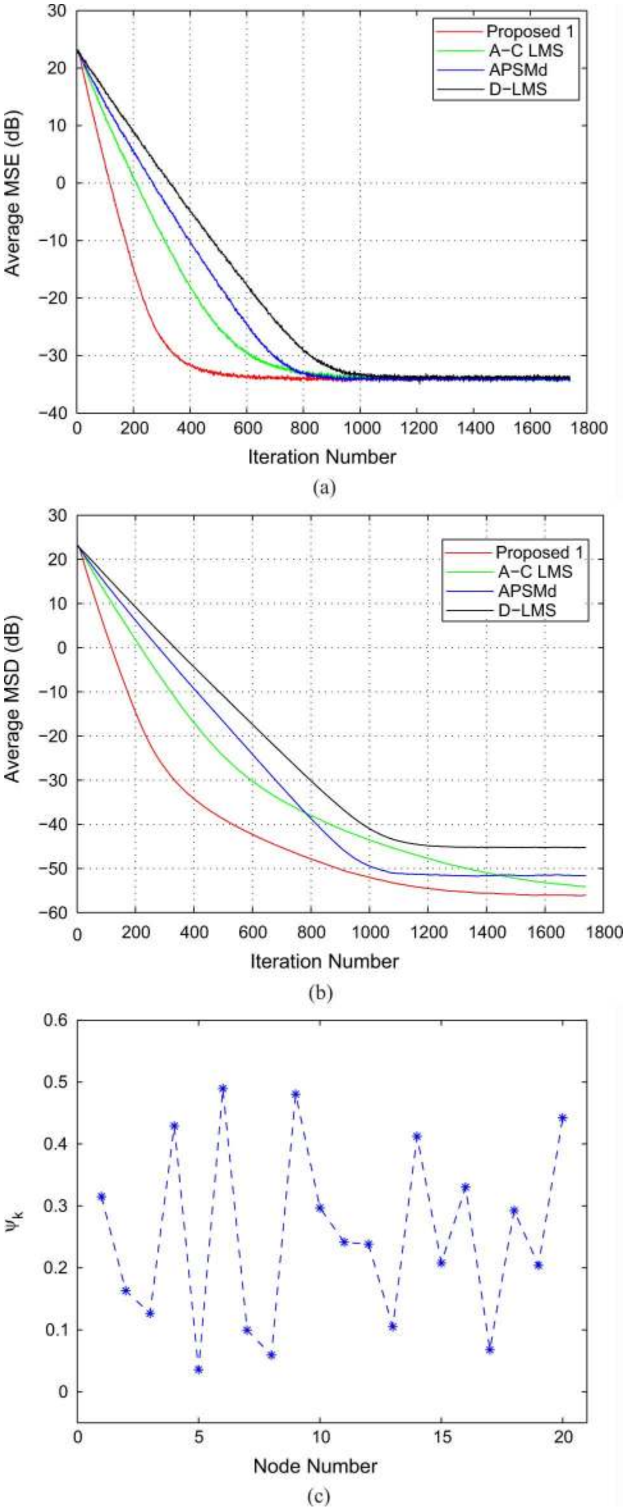


Fig. 5. (a) MSE for experiment 1. (b) MSD for experiment 1. (c) The statistics of the network's regressors.

Next, the algorithms are compared in a scenario where a subset of the nodes is malfunctioning. The number of nodes is chosen equal to $N = 10$ and the vector to be estimated is of dimension $m = 30$. Five of the nodes are malfunctioning, so for them $\sigma'_k = 40\sigma_k$, and $\sigma_u = 0.01$. For algorithm 2, $\tilde{\gamma}_k = 4\sigma_k$, $\hat{\gamma}_k = 2\tilde{\gamma}_k$, $\tilde{c}_k = 5\tilde{\gamma}_k$, $\hat{c}_k = 5\hat{\gamma}_k$. Finally, the rest of the parameters are $\psi_u = 0.5$, $q = 8$ in the projection based algorithms, $\mu_{LMS} = 0.08$ and $\mu_{D-LMS} = 0.01$. The stepsizes, as in the previous experiments, were chosen so that the algorithms converge to the same steady state error floor, in the MSE sense.

From Fig. 8(a), it can be observed that the projection based algorithms converge faster to the common error floor. Furthermore, the proposed algorithm 2 and the APSMd have a similar convergence speed. In Fig. 8(c), the average MSE, taken over the healthy nodes only, is given. It can be seen that the

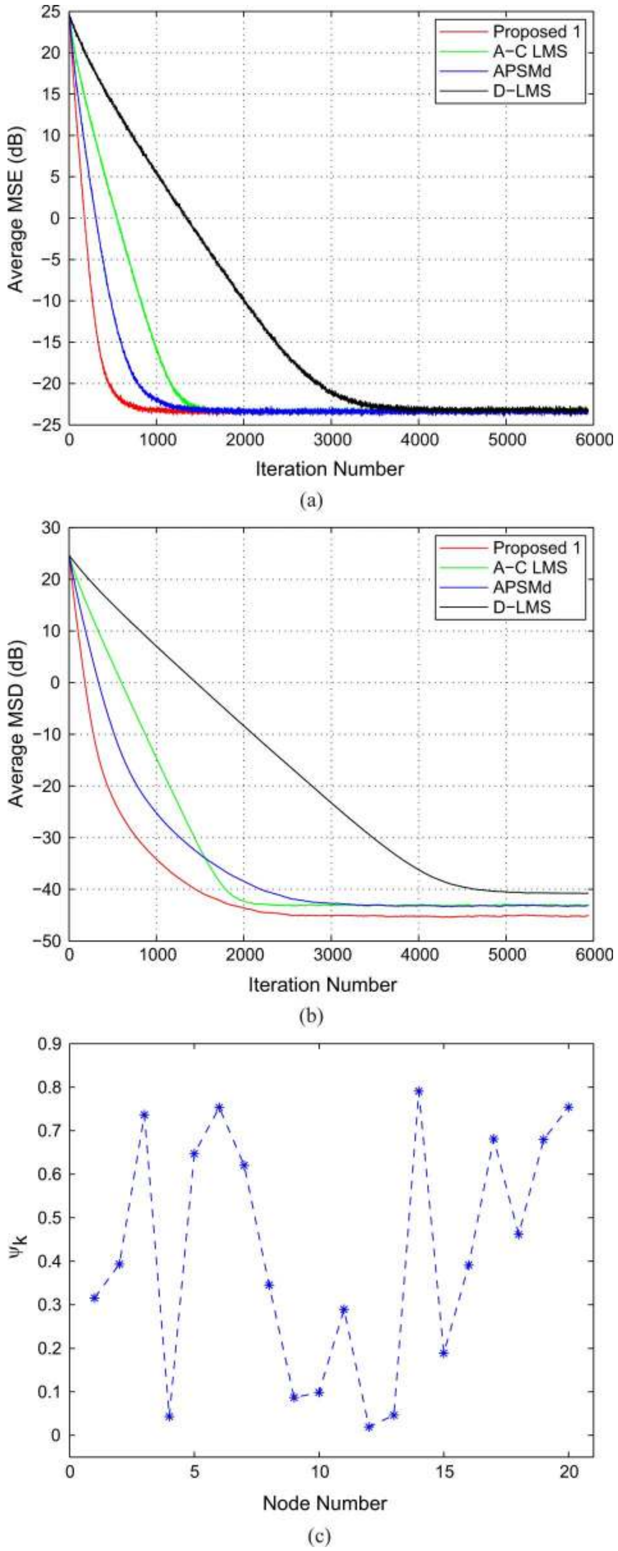


Fig. 6. (a) MSE for experiment 2. (b) MSD for experiment 2. (c) The statistics of the network's regressors.

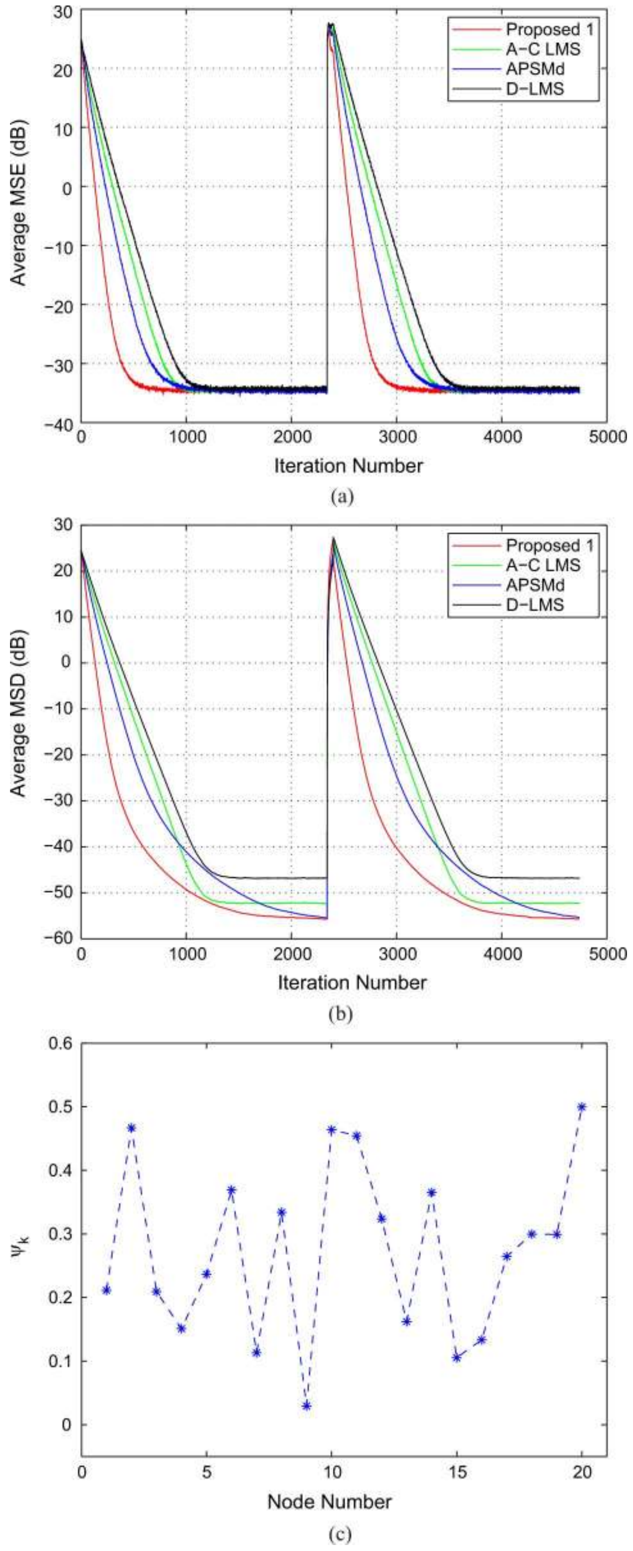
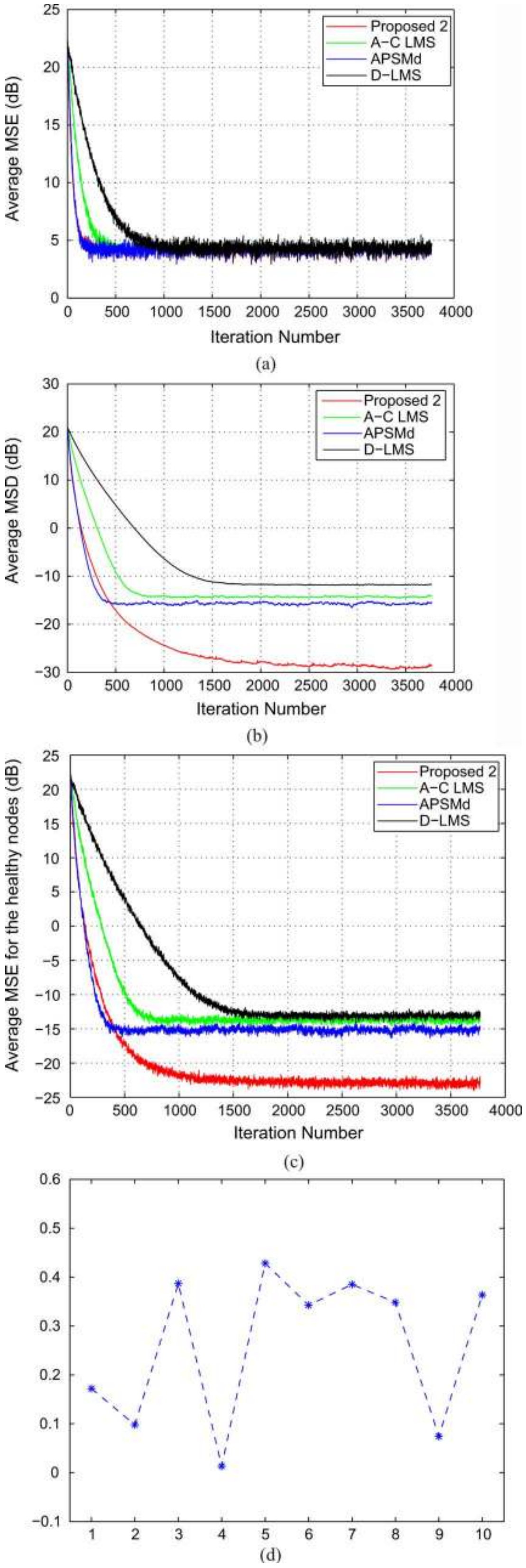


Fig. 7. (a) MSE for experiment 3. (b) MSD for experiment 3. (c) The statistics of the network's regressors.

proposed algorithm 2 exhibits a significantly better steady state error floor compared to the other algorithms. The reason that the proposed algorithm 2 achieves this improved error floor, whereas in Fig. 8(a) the algorithms converge to the same one, is a consequence of the fact that by taking into consideration the malfunctioning nodes, the noise dominates the average MSE. Furthermore, Fig. 8(b) demonstrates that the proposed algorithm 2 also achieves a significantly improved steady state error floor in the MSD sense, for the whole network. This implies that



(a)

(b)

(c)

(d)

Fig. 8. (a) MSE for experiment 4 by considering all the nodes of the network. (b) MSD for experiment 4. (c) Average MSE computed over the healthy nodes. (d) The statistics of the network's regressors.

the estimate occurring is closer to $\mathbf{w}^*$. The LMS-based algorithms result in the worst performance compared to the projection based algorithms, which is expected as in the APSMd and the proposed algorithm 2, robust cost functions are employed for minimization. However, the proposed algorithm 2 results in the lowest steady state error floor, since the Huber cost function accounts for the outliers, that occur due to the malfunctioning nodes, in a more focused way, compared to the hyperslabs used in APSMd. Finally, the fact that the proposed algorithm 2 converges slightly slower than APSMd, which can be seen from the curves of Fig. 8(b) and (c), is not a surprising result and it is due to the fact that in the case of hyperslabs the level set is reached with a single projection, whereas in the proposed algorithm 2, the corresponding level set is approached via a sequence of projections onto halfspaces that contain it.

VIII. Conclusion

Two novel efficient algorithms for distributed adaptive learning in diffusion networks have been developed. The schemes build upon convex analytic tools. A new combine-project-adapt protocol, where in every node the information collected by the neighborhood and the information sensed locally are "harmonized", is proposed. This is achieved by an extra projection that takes place after the combination of the received information, in every node, and before the adaptation step. The goal is to bring the result of the former step closer to the result of the latter one. Furthermore, the case where a number of nodes are malfunctioning was considered, by utilizing a Huber cost function, which is defined so as to take into consideration the presence of outliers. Full convergence results have been derived, while the stochastic analysis of this rich family of algorithms remains an open problem and is currently under investigation. The experiments verified the enhanced performance of the new algorithms compared to previously developed ones.

Appendix A
Mathematical Notation and Preliminaries

Basic Notions of Convex Analysis: A set $\mathcal{C} \subseteq \mathbb{R}^m$, for which it holds that $\forall \mathbf{b}_1, \mathbf{b}_2 \in \mathcal{C}$ and $\forall \alpha \in [0,1]$, $\alpha \mathbf{b}_1 + (1-\alpha)\mathbf{b}_2 \in \mathcal{C}$, is called convex. From a geometric point of view, this means that every line segment having as endpoints any $\mathbf{b}_1, \mathbf{b}_2$ will lie in $\mathcal{C}$. Moreover, a function $\Theta : \mathbb{R}^m \longrightarrow \mathbb{R}$ will be called convex if $\forall \mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^m$ and $\forall \alpha \in [0,1]$ the inequality $\Theta(\alpha \mathbf{b}_1 + (1-\alpha)\mathbf{b}_2) \leq \alpha \Theta(\mathbf{b}_1) + (1-\alpha)\Theta(\mathbf{b}_2)$ is satisfied. Finally, the subdifferential of Θ at an arbitrary point, say $\mathbf{b}$, is defined as the set of all subgradients of Θ at $\mathbf{b}$ ([30], [31]), i.e.

$$\partial \Theta(\mathbf{b}) := \{\mathbf{s} \in \mathbb{R}^m : \Theta(\mathbf{b}) + (\mathbf{x} - \mathbf{b})^T \mathbf{s} \leq \Theta(\mathbf{x}),\ \forall \mathbf{x} \in \mathbb{R}^m\}. \tag{21}$$

The subgradient of a convex function is a generalization of the gradient, which is only defined if the function is differentiable. As a matter of fact, if a convex function is differentiable, its subdifferential is a singleton, with a single element, i.e., the differential of the function at this point. It is well known that the differential at a point $\mathbf{b}$ has an elegant geometric interpretation. It defines the unique hyperplane, which is tangent at $\mathbf{b}$ to the graph of $\Theta(\mathbf{x})$. Moreover, if $\Theta(\mathbf{x})$ is convex, the graph of $\Theta(\mathbf{x})$ lies in one of the sides of this hyperplane. Similarly, each subgradient

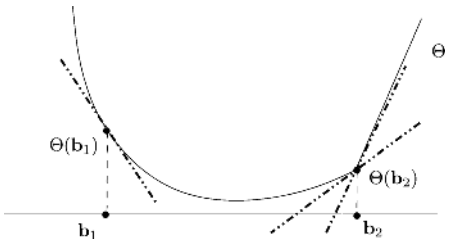


Fig. 9. Illustration of a gradient and two subgradients of a convex function. Note that in $\mathbf{b}_1$ the function is differentiable, so there exists a unique supporting hyperplane, which is tangent to the graph of the function, whereas in $\mathbf{b}_2$ the function is not differentiable and there exist more than one hyperplanes, that support the graph of the function.

at a point of a convex function is associated with a hyperplane that leaves the graph of $\Theta(\mathbf{x})$ on one of its side (supporting hyperplane). The only difference, now, is that there are more than one, possibly infinite, such hyperplanes. This is basically guaranteed by (21) and it is illustrated in Fig. 9.

The distance of an arbitrary point $\mathbf{b}$ from a closed non-empty convex set $\mathcal{C}$ is given by the distance function

$$\begin{aligned} d(\cdot,\mathcal{C}) : \mathbb{R}^m &\rightarrow [0,+\infty) \\ : \mathbf{b} &\mapsto \inf\{\|\mathbf{b}-\hat{\mathbf{b}}\| : \hat{\mathbf{b}} \in \mathcal{C}\}. \end{aligned}$$

This function is continuous, convex, nonnegative and is equal to zero for every point that lies in $\mathcal{C}$ [31]. Moreover, the projection mapping, $P_\mathcal{C}$ onto $\mathcal{C}$, is the mapping which takes a point $\mathbf{w}$ to the uniquely existing point, $P_\mathcal{C}(\mathbf{w}) \in \mathcal{C}$, such that

$$\|\mathbf{w} - P_\mathcal{C}(\mathbf{w})\| = d(\mathbf{w},\mathcal{C}).$$

It also holds that, $P_\mathcal{C}(\mathbf{b}) = \mathbf{b}$, $\forall \mathbf{b} \in \mathcal{C}$. In the sequel, we present some convex sets alongside their projection operators, that will be used throughout this paper.

Given a scalar d and a nonzero vector $\mathbf{u} \in \mathbb{R}^m$, the definition of a closed halfspace (Fig. 10(a)) is given by $H^- := \{\mathbf{w} \in \mathbb{R}^m : \mathbf{w}^T\mathbf{u} \leq d\}$ and the projection operator associated with it is given by $P_{H^-}(\mathbf{w}) = \mathbf{w} - \frac{\min\{0,\mathbf{w}^T\mathbf{u}-d\}}{\|\mathbf{u}\|^2}\mathbf{u}$, $\forall \mathbf{w} \in \mathbb{R}^m$. In a similar notion, we define the hyperplane $H := \{\mathbf{w} \in \mathbb{R}^m : \mathbf{w}^T\mathbf{u} = d\}$ and the resulting projection operator is $P_H(\mathbf{w}) := \mathbf{w} - \frac{\mathbf{w}^T\mathbf{u}-d}{\|\mathbf{u}\|^2}\mathbf{u}$, $\forall \mathbf{w} \in \mathbb{R}^m$.

Another typical example of a closed convex set is a hyperslab, which is illustrated in Fig. 10(b). For an $\epsilon > 0$, a hyperslab is defined as the set $S := \{\mathbf{w} \in \mathbb{R}^m : |d - \mathbf{w}^T\mathbf{u}| \leq \epsilon\}$, and the projection of a point onto it is given by

$$P_S(\mathbf{w}) = \begin{cases} \mathbf{w}, & \text{if } |d - \mathbf{w}^T\mathbf{u}| \leq \epsilon \\ \mathbf{w} + \frac{d-\mathbf{w}^T\mathbf{u}-\epsilon}{\|\mathbf{u}\|^2}\mathbf{u}, & \text{if } d - \mathbf{w}^T\mathbf{u} > \epsilon \\ \mathbf{w} + \frac{d-\mathbf{w}^T\mathbf{u}+\epsilon}{\|\mathbf{u}\|^2}\mathbf{u}, & \text{if } d - \mathbf{w}^T\mathbf{u} < -\epsilon. \end{cases}$$

This family of sets plays a central role in many applications, e.g., [20], [32], and [33], and are associated with loss functions that spring from the rationale of robust statistics [15], [25], and [28].

Two more convex sets, which will be used in our theoretical analysis of the algorithms, are the closed and open balls with center $\mathbf{c}$ and radius δ (Fig. 10(c)). The definition of the closed ball set is: $B_{[\mathbf{c},\delta]} := \{\mathbf{w} \in \mathbb{R}^m : \|\mathbf{w} - \mathbf{c}\| \leq \delta\}$. In a similar notion, the open ball is defined as $B_{(\mathbf{c},\delta)} := \{\mathbf{w} \in \mathbb{R}^m : \|\mathbf{w} - \mathbf{c}\| < \delta\}$.

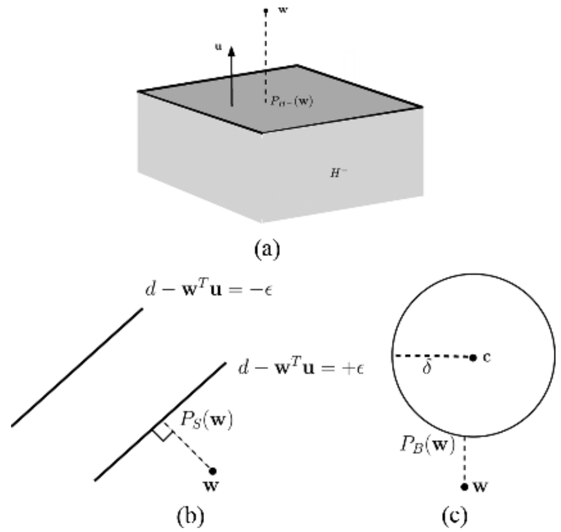

w
u
$P_{H^-}(\mathbf{w})$
H^-
(a)
$d - \mathbf{w}^T\mathbf{u} = -\epsilon$
$d - \mathbf{w}^T\mathbf{u} = +\epsilon$
$P_S(\mathbf{w})$
w
(b)
c
δ
$P_B(\mathbf{w})$
w
(c)

Fig. 10. (a) The geometry of a halfspace. Its boundary is a hyperplane. (b) A hyperslab and the projection of a point onto it. (c) A closed ball $B_{[\mathbf{c},\delta]}$.

Finally, the relative interior of a nonempty set, $\mathcal{C}$, with respect to another one, $\mathcal{S}$, is defined as

$$\mathrm{ri}_\mathcal{S}(\mathcal{C}) = \{\mathbf{w} \in \mathcal{C} : \exists \varepsilon_0 > 0 \text{ with } \emptyset \neq (B_{(\mathbf{w},\varepsilon_0)} \cap \mathcal{S}) \subset \mathcal{C}\}.$$

This will be used in the proof of Theorem 1.4.

Appendix B
Proof of Theorem 1

Proof of Theorem 1.1: Assume that for a fixed node, say k, at time instant $n+1$ we have estimated $\mathbf{z}_k(n)$. We define the cost function, for any $\mathbf{w} \in \mathbb{R}^m$[2]

$$\Theta_{k,n+1}(\mathbf{w}) = \begin{cases} \sum\limits_{j\in\mathcal{J}_{n+1}} \frac{\omega_{k,j} d(\mathbf{z}_k(n), S_{k,j})}{\sum\limits_{l\in\mathcal{J}_{n+1}} \omega_{k,l} d(\mathbf{z}_k(n), S_{k,l})} d(\mathbf{w}, S_{k,j}), \\ \qquad\qquad \text{if } \mathbf{z}_k(n) \notin \bigcap\limits_{j\in\mathcal{J}_{n+1}} S_{k,j} \\ 0, \qquad\qquad \text{otherwise.} \end{cases}$$

We also define $L_{n+1} := \sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} d(\mathbf{z}_k(n), S_{k,j})$, for which, by definition, $L_{n+1} \neq 0$ if $\mathbf{z}_k(n) \notin \bigcap_{j\in\mathcal{J}_{n+1}} S_{k,j}$. The previous statement holds true obviously because if $\mathbf{z}_k(n) \notin \bigcap_{j\in\mathcal{J}_{n+1}} S_{k,j}$ there exists $j_0 \in \mathcal{J}_{n+1}$ for which $d(\mathbf{z}_k(n), S_{k,j_0}) \neq 0$ hence $\sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} d(\mathbf{z}_k(n), S_{k,j}) > 0$.

It can be seen that this cost function is convex, continuous and subdifferentiable. Its subgradient is given by [26]

$$\Theta'_{k,n+1}(\mathbf{w}) = \begin{cases} \frac{1}{L_{n+1}} \sum\limits_{j\in\mathcal{J}_{n+1}} \omega_{k,j} d(\mathbf{z}_k(n), S_{k,j}) d'(\mathbf{w}, S_{k,j}) \\ \qquad\qquad \text{if } \mathbf{z}_k(n) \notin \bigcap\limits_{j\in\mathcal{J}_{n+1}} S_{k,j} \\ 0, \qquad\qquad \text{otherwise} \end{cases} \tag{22}$$

[2] The proof when we use projections onto halfspaces, which are involved when the Huber cost function is employed, follows similar steps. All one has to do is to modify the cost function replacing projections onto hyperslabs with projections onto halfspaces. However, the proofs rely on the properties of metric projections and not on their specific form.

where $d'(\mathbf{w}, S_{k,j}) \in \partial d(\mathbf{w}, S_{k,j})$, and [31]

$$\partial d(\mathbf{w}, S_{k,j}) = \begin{cases} \frac{\mathbf{w}-P_{S_{k,j}}(\mathbf{w})}{d(\mathbf{w},S_{k,j}(\mathbf{w}))}, & \text{if } \mathbf{w} \notin S_{k,j} \\ \text{a certain subset of } B_{[0,1]} \text{ which contains } \mathbf{0} & \\ & \text{otherwise.} \end{cases}$$

Additionally, if $\mathbf{w} \notin \bigcap_{j\in\mathcal{J}_{n+1}} S_{k,j}$ and $\mathbf{z}_k(n) \notin \bigcap_{j\in\mathcal{J}_{n+1}} S_{k,j}$, from (22) a choice of a subgradient is

$$\Theta'_{k,n+1}(\mathbf{w}) = \frac{1}{L_{n+1}} \sum_{\{j\in\mathcal{J}_{n+1}:\mathbf{w}\notin S_{k,j}\}} \omega_{k,j} d(\mathbf{z}_k(n), S_{k,j}) \frac{\mathbf{w} - P_{S_{k,j}}(\mathbf{w})}{d(\mathbf{w}, S_{k,j})}.$$

Finally, if $\mathbf{z}_k(n) \notin \bigcap_{j\in\mathcal{J}_{n+1}} S_{k,j}$

$$\begin{aligned} \Theta'_{k,n+1}(\mathbf{z}_k(n)) &= \frac{1}{L_{n+1}} \sum_{\{j\in\mathcal{J}_{n+1}:\mathbf{z}_k(n)\notin S_{k,j}\}} \omega_{k,j} \left(\mathbf{z}_k(n) - P_{S_{k,j}}(\mathbf{z}_k(n))\right) \\ &= \frac{1}{L_{n+1}} \sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} \left(\mathbf{z}_k(n) - P_{S_{k,j}}(\mathbf{z}_k(n))\right). \qquad (23) \end{aligned}$$

The last equality is true, due to the fact that if there exists $j_0 \in \mathcal{J}_{n+1}$ such that $\mathbf{z}_k(n) \in S_{k,j_0}$, then $\mathbf{z}_k(n) = P_{S_{k,j_0}}(\mathbf{z}_k(n))$.

Now, recall the definition of $\mathcal{M}_k(n+1)$ given in (13). We define

$$\lambda_k(n+1) = \frac{\mu_k(n+1)}{\mathcal{M}_k(n+1)}. \qquad (24)$$

One should notice here that $\lambda_k(n+1) \in (0,2)$ under assumption (2). In the case where $\sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j}\left(\mathbf{z}_k(n) - P_{S_{k,j}}(\mathbf{z}_k(n))\right) \neq \mathbf{0}$, if we go back to the recursion given in (12) and combining (24) with (22) and (23), we get

$$\begin{aligned} \mathbf{w}_k(n+1) &= \mathbf{z}_k(n) + \mu_k(n+1) \\ &\times \left(\sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_k(n)) - \mathbf{z}_k(n) \right) \\ &= \mathbf{z}_k(n) + \lambda_k(n+1) \frac{\sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} d^2(\mathbf{z}_k(n), S_{k,j})}{\left\| \sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j}\left(\mathbf{z}_k(n) - P_{S_{k,j}}(\mathbf{z}_k(n))\right)\right\|^2} \\ &\times \left(\sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} \left(P_{S_{k,j}}(\mathbf{z}_k(n)) - \mathbf{z}_k(n)\right)\right) \\ &= \mathbf{z}_k(n) + \lambda_k(n+1) \\ &\times \frac{\frac{1}{L_{n+1}} \sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} d^2(\mathbf{z}_k(n), S_{k,j})}{\frac{1}{L_{n+1}^2} \left\| \sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j}\left(\mathbf{z}_k(n) - P_{S_{k,j}}(\mathbf{z}_k(n))\right)\right\|^2} \frac{1}{L_{n+1}} \\ &\times \left(\sum_{j\in\mathcal{J}_{n+1}} \omega_{k,j} \left(P_{S_{k,j}}(\mathbf{z}_k(n)) - \mathbf{z}_k(n)\right)\right) \\ &= \mathbf{z}_k(n) - \lambda_k(n+1) \frac{\Theta_{k,n+1}(\mathbf{z}_k(n))}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} \Theta'_{k,n+1}(\mathbf{z}_k(n)). \qquad (25) \end{aligned}$$

Equation (25) is a slight modification of the adaptive projected subgradient method. Following similar steps as in [26], and under assumption (1), then it can be shown that: $\mathbf{z}_k(n) \notin \bigcap_{j\in\mathcal{J}_{n+1}} S_{k,j} \Leftrightarrow \|\Theta'_{k,n+1}(\mathbf{z}_k(n))\| \neq 0, \forall k \in \mathcal{N}, \forall n \geq n_0$. So for the whole network we have (26) and (27) shown at the bottom of the next page. Notice here that $P_{S'_{k,n+1}}(\hat{\mathbf{w}}) = \hat{\mathbf{w}}, \forall k \in \mathcal{N}$. This argument is true since $\hat{\mathbf{w}} \in \Omega(n) \Rightarrow \hat{\mathbf{w}} \in S_{k,n+1} \subset S'_{k,n+1} \Rightarrow \hat{\mathbf{w}} \in S'_{k,n+1}, \forall k \in \mathcal{N}$. Furthermore, one basic property of the projection operator onto closed convex sets [34] states that $\|P_{S'_{k,n+1}}(\boldsymbol{\phi}_k(n)) - P_{S'_{k,n+1}}(\hat{\mathbf{w}})\| \leq \|\boldsymbol{\phi}_k(n) - \hat{\mathbf{w}}\|, \forall k \in \mathcal{N}$. Hence, if we recall the properties of the Consensus matrix, we have

$$\begin{aligned} &\left\| \begin{bmatrix} \mathbf{z}_1(n) - \hat{\mathbf{w}} \\ \mathbf{z}_2(n) - \hat{\mathbf{w}} \\ \vdots \\ \mathbf{z}_N(n) - \hat{\mathbf{w}} \end{bmatrix} \right\|^2 \\ &= \left\| \begin{bmatrix} P_{S'_{1,n+1}}(\boldsymbol{\phi}_1(n)) - P_{S'_{1,n+1}}(\hat{\mathbf{w}}) \\ P_{S'_{2,n+1}}(\boldsymbol{\phi}_2(n)) - P_{S'_{2,n+1}}(\hat{\mathbf{w}}) \\ \vdots \\ P_{S'_{N,n+1}}(\boldsymbol{\phi}_N(n)) - P_{S'_{N,n+1}}(\hat{\mathbf{w}}) \end{bmatrix} \right\|^2 \\ &\leq \left\| \begin{bmatrix} \boldsymbol{\phi}_1(n) - \hat{\mathbf{w}} \\ \boldsymbol{\phi}_2(n) - \hat{\mathbf{w}} \\ \vdots \\ \boldsymbol{\phi}_N(n) - \hat{\mathbf{w}} \end{bmatrix} \right\|^2 = \|\mathbf{G}(n+1)\mathbf{w}(n) - \underline{\hat{\mathbf{w}}}\|^2 \\ &= \|\mathbf{G}(n+1)\mathbf{w}(n) - \mathbf{G}(n+1)\underline{\hat{\mathbf{w}}}\|^2 \\ &\leq \|\mathbf{G}(n+1)\|^2 \|\mathbf{w}(n) - \underline{\hat{\mathbf{w}}}\|^2 = \|\mathbf{w}(n) - \underline{\hat{\mathbf{w}}}\|^2. \qquad (28) \end{aligned}$$

Moreover, from the definition of the subgradient $\Theta'^{T}_{k,n+1}(\mathbf{z}_k(n))(\mathbf{z}_k(n) - \hat{\mathbf{w}}) \geq \Theta_{k,n+1}(\mathbf{z}_k(n)) - \Theta_{k,n+1}(\hat{\mathbf{w}}) = \Theta_{k,n+1}(\mathbf{z}_k(n)) \geq 0$, where we have used the fact that $\Theta_{k,n+1}(\hat{\mathbf{w}}) = 0$, which holds by the definition of the cost function, since $\hat{\mathbf{w}} \in \bigcap_{j\in\mathcal{J}_{n+1}} S_{k,j}$. Combining the last argument with (27) and (28), we have

$$\begin{aligned} &\|\mathbf{w}(n+1) - \underline{\hat{\mathbf{w}}}\|^2 \leq \|\mathbf{w}(n) - \underline{\hat{\mathbf{w}}}\|^2 \\ &+ \sum_{k=1}^{N} \lambda_k^2(n+1) \frac{(\Theta_{k,n+1}(\mathbf{z}_k(n)))^2}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} \\ &- 2\sum_{k=1}^{N} \lambda_k(n+1) \frac{(\Theta_{k,n+1}(\mathbf{z}_k(n)))^2}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} \\ &= \|\mathbf{w}(n) - \underline{\hat{\mathbf{w}}}\|^2 \\ &- \sum_{k=1}^{N} \lambda_k(n+1)(2 - \lambda_k(n+1)) \frac{(\Theta_{k,n+1}(\mathbf{z}_k(n)))^2}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} \\ &\Leftrightarrow \|\mathbf{w}(n) - \underline{\hat{\mathbf{w}}}\|^2 - \|\mathbf{w}(n+1) - \underline{\hat{\mathbf{w}}}\|^2 \\ &\geq \sum_{k=1}^{N} \lambda_k(n+1)\left(2 - \lambda_k(n+1)\right) \frac{(\Theta_{k,n+1}(\mathbf{z}_k(n)))^2}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} \geq 0. \end{aligned}$$

(29)

Hence, we conclude that $\|\mathbf{w}(n+1)-\underline{\hat{\mathbf{w}}}\| \leq \|\mathbf{w}(n)-\underline{\hat{\mathbf{w}}}\|$. This completes the proof of Theorem 1.1.

Remark 4: Here, we would like to point out that monotonicity also holds in cases where the subgradients of some nodes equal to zero. Assume, without loss of generality, that the subgradient of node l, at time instance $n+1$, is zero, which under assumption (1) implies that $\mathbf{z}_l(n) \in \bigcap_{j\in\mathcal{J}_{n+1}} S_{l,j}$. Then, from [26] and if assumption (1) holds true, it can be proved that the recursion for this node is $\mathbf{w}_l(n+1)=\mathbf{z}_l(n)$. Loosely speaking, the second term of the right hand side in (25) is omitted. So, following exactly similar steps as in the previous proof, (29) becomes $\|\mathbf{w}(n)-\underline{\hat{\mathbf{w}}}\| - \|\mathbf{w}(n+1)-\underline{\hat{\mathbf{w}}}\| \geq \sum_{k\in\mathcal{N}\setminus l}\lambda_k(n+1)\,(2-\lambda_k(n+1))\,\frac{(\Theta_{k,n+1}(\mathbf{z}_k(n)))^2}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} \geq 0$.

Proof of Theorem 1.2: We want to show that $\lim_{n\longrightarrow\infty} d(\mathbf{w}_k(n+1),\Omega(n)) = 0$. The sequence $\|\mathbf{w}(n)-\underline{\hat{\mathbf{w}}}\|^2$ is bounded and monotonically decreasing, hence it converges. So it is a Cauchy sequence, from which we obtain that

$$\|\mathbf{w}(n)-\underline{\hat{\mathbf{w}}}\|^2 - \|\mathbf{w}(n+1)-\underline{\hat{\mathbf{w}}}\|^2 \longrightarrow 0,\ n\longrightarrow\infty. \tag{30}$$

So from (29) and (30)

$$\lim_{n\longrightarrow\infty}\lambda_k(n+1)(2-\lambda_k(n+1))\frac{(\Theta_{k,n+1}(\mathbf{z}_k(n)))^2}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} = 0,\ \forall k\in\mathcal{N}. \tag{31}$$

If we make one more assumption that states that $\Theta'_{k,n+1}(\mathbf{z}_k(n))$ is bounded[3], i.e., $\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\| \leq D, \forall k\in\mathcal{N},\ \forall n\in\mathbb{N}$, under assumption (3) and if we take into consideration (31), we have that

$$\frac{\varepsilon_1^2}{D^2}\Theta_{k,n+1}(\mathbf{z}_k(n)) \leq \frac{\lambda_k(n+1)(2-\lambda_k(n+1))}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2} \times\Theta_{k,n+1}(\mathbf{z}_k(n)) \longrightarrow 0, \forall k\in\mathcal{N}.$$

From the last equation, we have that

$$\lim_{n\longrightarrow\infty}\Theta_{k,n+1}(\mathbf{z}_k(n)) = 0,\ \forall k\in\mathcal{N}. \tag{32}$$

Now, if we go back to the recursion given in (26) and combine it with (31) we obtain

$$\lim_{n\longrightarrow\infty}\|\mathbf{w}_k(n+1)-\mathbf{z}_k(n)\| = 0,\ \forall k\in\mathcal{N}. \tag{33}$$

Let us assume that $\mathbf{v}$ is an arbitrary point that belongs to $\Omega(n)$. We have that $\|\mathbf{w}_k(n+1)-\mathbf{v}\| \leq \|\mathbf{w}_k(n+1)-\mathbf{z}_k(n)\| + \|\mathbf{z}_k(n)-\mathbf{v}\|$, where this holds due to the triangle inequality. Therefore

$$\begin{aligned}\inf_{\mathbf{v}\in\Omega(n)}\|\mathbf{w}_k(n+1)-\mathbf{v}\| &\leq \|\mathbf{w}_k(n+1)-\mathbf{z}_k(n)\| \\ &\quad+\inf_{\mathbf{v}\in\Omega(n)}\|\mathbf{z}_k(n)-\mathbf{v}\| \\ \Leftrightarrow d(\mathbf{w}_k(n+1),\Omega(n)) &\leq \|\mathbf{w}_k(n+1)-\mathbf{z}_k(n)\| \\ &\quad+ d(\mathbf{z}_k(n),\Omega(n)).\end{aligned} \tag{34}$$

Following the same steps as in [26] and taking into consideration (32), it can be proved that

$$\lim_{n\longrightarrow\infty} d(\mathbf{z}_k(n),\Omega(n)) = 0. \tag{35}$$

$$\mathbf{w}(n+1) = \begin{bmatrix} \mathbf{z}_1(n) - \lambda_1(n+1)\frac{\Theta_{1,n+1}(\mathbf{z}_1(n))}{\left\|\Theta'_{1,n+1}(\mathbf{z}_1(n))\right\|^2}\Theta'_{1,n+1}(\mathbf{z}_1(n)) \\ \mathbf{z}_2(n) - \lambda_2(n+1)\frac{\Theta_{2,n+1}(\mathbf{z}_2(n))}{\left\|\Theta'_{2,n+1}(\mathbf{z}_2(n))\right\|^2}\Theta'_{2,n+1}(\mathbf{z}_2(n)) \\ \vdots \\ \mathbf{z}_N(n) - \lambda_N(n+1)\frac{\Theta_{N,n+1}(\mathbf{z}_N(n))}{\left\|\Theta'_{N,n+1}(\mathbf{z}_N(n))\right\|^2}\Theta'_{N,n+1}(\mathbf{z}_N(n)) \end{bmatrix} \Rightarrow \tag{26}$$

$$\begin{aligned}\|\mathbf{w}(n+1)-\underline{\hat{\mathbf{w}}}\|^2 &= \left\|\begin{bmatrix} \mathbf{z}_1(n) - \lambda_1(n+1)\frac{\Theta_{1,n+1}(\mathbf{z}_1(n))}{\left\|\Theta'_{1,n+1}(\mathbf{z}_1(n))\right\|^2}\Theta'_{1,n+1}(\mathbf{z}_1(n)) \\ \mathbf{z}_2(n) - \lambda_2(n+1)\frac{\Theta_{2,n+1}(\mathbf{z}_2(n))}{\left\|\Theta'_{2,n+1}(\mathbf{z}_2(n))\right\|^2}\Theta'_{2,n+1}(\mathbf{z}_2(n)) \\ \vdots \\ \mathbf{z}_N(n) - \lambda_N(n+1)\frac{\Theta_{N,n+1}(\mathbf{z}_N(n))}{\left\|\Theta'_{N,n+1}(\mathbf{z}_N(n))\right\|^2}\Theta'_{N,n+1}(\mathbf{z}_N(n)) \end{bmatrix} - \underline{\hat{\mathbf{w}}}\right\|^2 \\ &= \left\|\begin{bmatrix}\mathbf{z}_1(n)-\hat{\mathbf{w}} \\ \mathbf{z}_2(n)-\hat{\mathbf{w}} \\ \vdots \\ \mathbf{z}_N(n)-\hat{\mathbf{w}}\end{bmatrix}\right\|^2 + \sum_{k=1}^{N}\lambda_k^2(n+1)\frac{(\Theta_{k,n+1}(\mathbf{z}_k(n)))^2}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^4}\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2 \\ &\quad - 2\sum_{k=1}^{N}\lambda_k(n+1)\frac{\Theta_{k,n+1}(\mathbf{z}_k(n))\Theta'^{T}_{k,n+1}(\mathbf{z}_k(n))(\mathbf{z}_k(n)-\hat{\mathbf{w}})}{\left\|\Theta'_{k,n+1}(\mathbf{z}_k(n))\right\|^2}.\end{aligned} \tag{27}$$

[3] This assumption is realistic in general and it is adopted in the analysis of the APSM related algorithms.

Taking limits into (34), and combining (33), (35) we conclude that:

$$\lim_{n\longrightarrow\infty} d(\mathbf{w}_k(n+1),\Omega(n))=0.$$

Proof of Theorem 1.3: First we need to prove the following claim.

Claim 1: Assume that $\epsilon'_k>\epsilon_k$ and that (1), (3) hold true. Then, there exists n_2 such that $\boldsymbol{\phi}_k(n)\in S'_{k,n+1}, \forall n\geq n_2, \forall k\in\mathcal{N}$.

Proof: Since $\epsilon'_k>\epsilon_k$, for any vector $\mathbf{w}$ on the boundary of $S'_{k,n+1}$, there exists ε_2, which depends on the choice of ϵ'_k and ϵ_k, such that $d(\mathbf{w},S_{k,n+1})>\varepsilon_2$. We have shown that $d(\mathbf{z}_k(n),\Omega(n))\longrightarrow 0$, $n\longrightarrow\infty$ under assumptions (1), (3). However, since by definition $\Omega(n)\subseteq S_{k,n+1}$ we have that $d(\mathbf{z}_k(n),S_{k,n+1})\longrightarrow 0$, $n\longrightarrow\infty$. Due to the last argument, there exists n_2 such that

$$d(\mathbf{z}_k(n),S_{k,n+1})\leq\frac{\varepsilon_2}{2},\ \forall n\geq n_2. \tag{36}$$

However, if $\exists n\geq n_2$ such that $\boldsymbol{\phi}_k(n)\notin S'_{k,n+1}$ then $\mathbf{z}_k(n)$ will lie on the boundary of $S'_{k,n+1}$, as it is the projection of $\boldsymbol{\phi}_k(n)$ onto it. Hence $d(\mathbf{z}_k(n),S_{k,n+1})>\varepsilon_2$ which clearly contradicts (36). Thus our claim holds true.

The fact that $\boldsymbol{\phi}_k(n)\in S'_{k,n+1}$, after some iterations, implies that $\mathbf{z}_k(n)=P_{S'_{k,n+1}}(\boldsymbol{\phi}_k(n))=\boldsymbol{\phi}_k(n), \forall n\geq n_2$. Recalling (26) we have $\forall n\geq n_2$

$$\begin{aligned}&\mathbf{w}(n+1)\\&=\begin{bmatrix}\boldsymbol{\phi}_1(n)-\lambda_1(n+1)\frac{\Theta_{1,n+1}(\boldsymbol{\phi}_1(n))}{\|\Theta'_{1,n+1}(\boldsymbol{\phi}_1(n))\|^2}\Theta'_{1,n+1}(\boldsymbol{\phi}_1(n))\\ \boldsymbol{\phi}_2(n)-\lambda_2(n+1)\frac{\Theta_{2,n+1}(\boldsymbol{\phi}_2(n))}{\|\Theta'_{2,n+1}(\boldsymbol{\phi}_2(n))\|^2}\Theta'_{2,n+1}(\boldsymbol{\phi}_2(n))\\ \vdots\\ \boldsymbol{\phi}_N(n)-\lambda_N(n+1)\frac{\Theta_{N,n+1}(\boldsymbol{\phi}_N(n))}{\|\Theta'_{N,n+1}(\boldsymbol{\phi}_N(n))\|^2}\Theta'_{N,n+1}(\boldsymbol{\phi}_N(n))\end{bmatrix}\\&=\mathbf{G}(n+1)\mathbf{w}(n)-\boldsymbol{F}(n+1)\end{aligned} \tag{37}$$

where

$$\begin{aligned}&\boldsymbol{F}(n+1)\\&=\begin{bmatrix}\lambda_1(n+1)\frac{\Theta_{1,n+1}(\boldsymbol{\phi}_1(n))}{\|\Theta'_{1,n+1}(\boldsymbol{\phi}_1(n))\|^2}\Theta'_{1,n+1}(\boldsymbol{\phi}_1(n))\\ \lambda_2(n+1)\frac{\Theta_{2,n+1}(\boldsymbol{\phi}_2(n))}{\|\Theta'_{2,n+1}(\boldsymbol{\phi}_2(n))\|^2}\Theta'_{2,n+1}(\boldsymbol{\phi}_2(n))\\ \vdots\\ \lambda_N(n+1)\frac{\Theta_{N,n+1}(\boldsymbol{\phi}_N(n))}{\|\Theta'_{N,n+1}(\boldsymbol{\phi}_N(n))\|^2}\Theta'_{N,n+1}(\boldsymbol{\phi}_N(n))\end{bmatrix}.\end{aligned}$$

Taking into consideration (31) we have that

$$\begin{aligned}&\lim_{n\longrightarrow\infty}\|\boldsymbol{F}(n+1)\|^2\\&=\lim_{n\longrightarrow\infty}\sum_{k=1}^{N}\|\lambda_k(n+1)\frac{\Theta_{k,n+1}(\boldsymbol{\phi}_k(n))}{\left\|\Theta'_{k,n+1}(\boldsymbol{\phi}_k(n))\right\|^2}\Theta'_{k,n+1}(\boldsymbol{\phi}_k(n))\|^2\\&=0.\end{aligned} \tag{38}$$

Now, going back and iterating (37), $\forall n\geq n_2$ we have

$$\begin{aligned}\mathbf{w}(n+1)&=\mathbf{G}(n+1)\mathbf{w}(n)-\boldsymbol{F}(n+1)\\&=\mathbf{G}(n+1)\mathbf{G}(n)\mathbf{w}(n-1)\\&\quad-\mathbf{G}(n+1)\boldsymbol{F}(n)-\boldsymbol{F}(n+1)\\&=\ldots=\prod_{j=0}^{n-n_2}\mathbf{G}(n-j+1)\mathbf{w}(n_2)\\&\quad-\sum_{j=n_2+1}^{n}\prod_{l=0}^{n-j}\mathbf{G}(n-l+1)\boldsymbol{F}(j)-\boldsymbol{F}(n+1).\end{aligned}$$

If we left-multiply the previous equation by $(\mathbf{I}_{Nm}-\mathbf{B}\mathbf{B}^T)$, we obtain

$$\begin{aligned}&(\mathbf{I}_{Nm}-\mathbf{B}\mathbf{B}^T)\mathbf{w}(n+1)\\&\quad=(\mathbf{I}_{Nm}-\mathbf{B}\mathbf{B}^T)\prod_{j=n_2}^{n}\mathbf{G}(n-j+1)\mathbf{w}(n_2)\\&\qquad-(\mathbf{I}_{Nm}-\mathbf{B}\mathbf{B}^T)\sum_{j=n_2+1}^{n}\prod_{l=0}^{n-j}\mathbf{G}(n-l+1)\boldsymbol{F}(j)\\&\qquad-(\mathbf{I}_{Nm}-\mathbf{B}\mathbf{B}^T)\boldsymbol{F}(n+1).\end{aligned}$$

If we exploit the decomposition of a consensus matrix, given in (4), and follow similar steps as in [19, Lemma 2] it can be verified that

$$\lim_{n\longrightarrow\infty}(\mathbf{I}_{Nm}-\mathbf{B}\mathbf{B}^T)\mathbf{w}(n+1)=\mathbf{0}, \tag{39}$$

which completes our proof.

Remark 5: Note that the result of this theorem can be readily generalized to the algorithm using the Huber loss function. The only condition needed to guarantee asymptotic consensus is $\mathrm{lev}_{\leq 0}\tilde{\Theta}_{k,n+1}\subset\mathrm{lev}_{\leq 0}\hat{\Theta}_{k,n+1}$, which by construction of the loss functions is true. □

Proof of Theorem 1.4: Recall that the projection operator, of an arbitrary vector $\mathbf{x}\in\mathbb{R}^{Nm}$ onto the consensus subspace equals to $P_{\mathcal{O}}(\mathbf{x})=\mathbf{B}\mathbf{B}^T\mathbf{x}, \forall\mathbf{x}\in\mathbb{R}^{Nm}$. Let assumptions a), c), d), and e) hold. Since assumption (e) holds, together with (16), from [11, Lemma 1] we have that there exists $\hat{\mathbf{w}}_*\in\mathcal{O}$ such that

$$\lim_{n\longrightarrow\infty}P_{\mathcal{O}}(\mathbf{w}(n))=\hat{\mathbf{w}}_*. \tag{40}$$

Now, exploiting the triangle inequality we have that

$$\begin{aligned}\|\mathbf{w}(n)-\hat{\mathbf{w}}_*\|&\leq\|\mathbf{w}(n)-P_{\mathcal{O}}(\mathbf{w}(n))\|\\&\quad+\|\hat{\mathbf{w}}_*-P_{\mathcal{O}}(\mathbf{w}(n))\|\longrightarrow 0,\ n\longrightarrow\infty\end{aligned} \tag{41}$$

where this limit holds from (39) and (40). The proof is complete since (41) implies that $\lim_{n\longrightarrow\infty}\mathbf{w}(n)=\hat{\mathbf{w}}_*$.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, 2002.

[2] D. Blatt and A. Hero, "Energy based sensor network source localization via projection onto convex sets (POCS)," *IEEE Trans. Signal Processing*, vol. 54, no. 9, pp. 3614–3619, 2006.

[3] D. Li, K. Wong, Y. Hu, and A. Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, 2002.

[4] X. Nguyen, M. Jordan, and B. Sinopoli, "A kernel-based learning approach to ad hoc sensor network localization," *ACM Trans. Sensor Networks (TOSN)*, vol. 1, no. 1, pp. 134–152, 2005.

[5] T. Sheng-Yuan and A. H. Sayed, "Foraging behavior of fish schools via diffusion adaptation," in *Proc. 2010 IAPR Workshop on Cognitive Inf. Processing*, Elba Island, Italy, 2010, pp. 63–68.

[6] C. Lopes and A. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Processing*, vol. 55, no. 8, pp. 4064–4077, 2007.

[7] L. Li, J. Chambers, C. Lopes, and A. Sayed, "Distributed estimation over an adaptive incremental network based on the affine projection algorithm," *IEEE Trans. Signal Processing*, vol. 58, no. 1, pp. 151–164, 2009.

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

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

[10] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in ad hoc wsns with noisy links-part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[11] I. Yamada and N. Ogura, "Adaptive projected subgradient method for asymptotic minimization of sequence of nonnegative convex functions," *Num. Funct. Anal. Optim.*, vol. 25, no. 7–8, pp. 593–617, 2004.

[12] K. Slavakis, I. Yamada, and N. Ogura, "The adaptive projected subgradient method over the fixed point set of strongly attracting nonexpansive mappings," *Num. Funct. Anal. Optim.*, vol. 27, no. 8, pp. 905–930, 2006.

[13] K. Slavakis and I. Yamada, "Asymptotic minimization of sequences of loss functions constrained by families of quasi-nonexpansive mappings and its application to online learning," *Arxiv Preprint ArXiv:1008.5231*, 2010.

[14] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Processing Mag.*, vol. 28, no. 1, pp. 97–123, 2011.

[15] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proc. ACM 3rd Int. Symp. Inf. Processing in Sensor Networks*, New York, NY, 2004, pp. 20–27.

[16] A. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ: Wiley, 2003.

[17] P. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. New York: Springer Verlag, 2008.

[18] G. Mateos, I. Schizas, and G. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP J. Adv. Signal Processing*, vol. 2009, 2010.

[19] R. Cavalcante, I. Yamada, and B. Mulgrew, "An adaptive projected subgradient approach to learning in diffusion networks," *IEEE Trans. Signal Processing*, vol. 57, pp. 2762–2774, 2009.

[20] S. Chouvardas, K. Slavakis, and S. Theodoridis, "A novel adaptive algorithm for diffusion networks using projections onto hyperslabs," in *Proc. IAPR Workshop Cognitive Inf. Processing*, Elba Island, Italy, 2010, pp. 393–398.

[21] S. Stankovic, M. Stankovic, and D. Stipanovic, "Decentralized parameter estimation by consensus based stochastic approximation," in *Proc. 46th IEEE Conf. Decision Contr.*, 2007, pp. 1535–1540.

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

[23] Y. Lu, V. Roychowdhury, and L. Vandenberghe, "Distributed parallel support vector machines in strongly connected networks," *IEEE Trans. Neural Networks*, vol. 19, no. 7, pp. 1167–1178, 2008.

[24] B. Polyak, "Minimization of unsmooth functionals," *USSR Comput. Math. Math. Phys.*, vol. 9, no. 3, pp. 14–29, 1969.

[25] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th ed. New York: Academic, 2009.

[26] K. Slavakis, S. Theodorids, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Processing*, vol. 56, no. 7, pt. 1, pp. 2781–2796, 2008.

[27] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. 44th IEEE Conf. European Contr. CDC-ECC'05*, 2005, pp. 2996–3000.

[28] P. Huber and E. Ronchetti, *Robust Statistics*. Hoboken, NJ: Wiley, 2009.

[29] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Adaptive multiregression in reproducing kernel Hilbert spaces: The multiaccess MIMO channel case," *IEEE Trans. Neural Netw.*, 2010, with minor revisions, accepted for publication.

[30] D. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex Analysis and Optimization*. New York: Athena Scientific, 2003.

[31] J. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms: Fundamentals*. New York: Springer, 1993.

[32] K. Slavakis, S. Theodoridis, and I. Yamada, "Adaptive constrained learning in reproducing kernel hilbert spaces: The robust beamforming case," *IEEE Trans. Signal Processing*, vol. 57, no. 12, pp. 4744–4764, 2009.

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

[34] H. Stark, Y. Yang, and Y. Yang, *Vector Space Projections: A Numerical Approach to Signal and Image Processing, Neural Nets, and Optics*. New York: Wiley, 1998.

Symeon Chouvardas (S'11) received the B.Sc. degree from the Department of Informatics and Telecommunications, University of Athens, Athens, Greece, in 2008. He is currently pursuing the Ph.D. degree in the Department of Informatics and Telecommunications, University of Athens.

His research interests are in adaptive filter theory, machine learning and distributed learning.

Konstantinos Slavakis (M'06) received the M.E. and Ph.D. degrees in electrical and electronic engineering from Tokyo Institute of Technology (TokyoTech), Tokyo, Japan, in 1999 and 2002, respectively.

From 1996 to 2002, he was a recipient of the Japanese Government (MEXT) Scholarship. From 2004 to 2006, he was with TokyoTech as a Japan Society for the Promotion of Science (JSPS) Postdoctoral Fellow, and from 2006 to 2007, he was a Postdoctoral Fellow in the Department of Informatics and Telecommunications, University of Athens, Greece. Since September 2007, he has been an Assistant Professor for the Department of Telecommunications Science and Technology, University of Peloponnese, Tripolis, Greece. His research interests include the applications of convex analysis and computational algebraic geometry to signal processing, machine learning, array, and multidimensional systems problems.

Dr. Slavakis serves as an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.

Sergios Theodoridis (M'87–SM'02–F'08) is currently Professor of Signal Processing and Communications in the Department of Informatics and Telecommunications of the University of Athens. His research interests lie in the areas of adaptive algorithms and communications, machine learning and pattern recognition, signal processing for audio processing and retrieval. He is the coeditor of the book *Efficient Algorithms for Signal Processing and System Identification* (Prentice Hall, 1993), the coauthor of the best selling book *Pattern Recognition* (Academic, 4th ed., 2008), the coauthor of the book *Introduction to Pattern Recognition: A MATLAB Approach* (Academic, 2009), and the coauthor of three books in Greek, two of them for the Greek Open University.

Dr. Theodoridis is the coauthor of six papers that have received best paper awards including the 2009 IEEE Computational Intelligence Society Transactions on Neural Networks Outstanding paper Award. He has served as an IEEE Signal Processing Society Distinguished Lecturer. He was the General Chairman of EUSIPCO-98, the Technical Program Co-Chair for ISCAS-2006 and Co-Chairman and Co-Founder of CIP-2008, and Co-Chairman of CIP-2010. He has served as President of the European Association for Signal Processing (EURASIP) and as member of the Board of Governors for the IEEE CAS Society. He currently serves as member of the Board of Governors (Member-at-Large) of the IEEE SP Society. He has served as a member of the Greek National Council for Research and Technology and he was Chairman of the SP advisory committee for the Edinburgh Research Partnership (ERP). He has served as Vice chairman of the Greek Pedagogical Institute and he was for four years member of the Board of Directors of COSMOTE (the Greek mobile phone operating company). He is Fellow of IET, a Corresponding Fellow of RSE, a Fellow of EURASIP and a Fellow of IEEE.