ADAPTIVE ESTIMATION ALGORITHMS OVER DISTRIBUTED NETWORKS

Ali H. Sayed and Cassio G. Lopes

Department of Electrical Engineering
University of California
Los Angeles, CA, 90095

ABSTRACT

We provide an overview of adaptive estimation algorithms over distributed networks. The algorithms rely on local collaborations and exploit the space-time structure of the data. Each node is allowed to communicate with its neighbors in order to exploit the spatial dimension, while it also evolves locally to account for the time dimension. Algorithms of the least-mean-squares and least-squares types are described. Both incremental and diffusion strategies are considered.

1. INTRODUCTION

Distributed networks linking PCs, laptops, cell phones, sensors and actuators will form the backbone of future data communication and control networks. Applications will range from sensor networks to precision agriculture, environment monitoring, disaster relief management, smart spaces, target localization, as well as medical applications [1]–[4]. In all these cases, the distribution of the nodes in the field yields spatial diversity, which should be exploited alongside the temporal dimension in order to enhance the robustness of the processing tasks and improve the probability of signal and event detection [1].

Distributed processing deals with the extraction of information from data collected at nodes that are distributed over a geographic area. For example, each node in a network of nodes could collect noisy observations related to a certain parameter of interest. The nodes would then interact with each other in a certain manner, as dictated by the network topology, in order to arrive at an estimate of the parameter. The objective is to arrive at an estimate that is as accurate as the one that would be obtained if each node had access to the information across the entire network. [1, 5].

1.1. Example

Consider a collection of N nodes spread over a geographic area, as shown in Fig. 1. Each node has access to a local temperature measurement T_i and the objective is to provide the nodes with information about the average temperature $\bar{T}$ across the network.

In a centralized solution, the nodes would send their temperature measurements to a central processor, which would then average these measurements and compute $\bar{T}$. The average temperature would subsequently be communicated back to the nodes. In one distributed solution (known as a consensus implementation) each node combines the measurements from its immediate neighbors (those that are connected to it). The result of the combination becomes this node's new measurement. For example, for node 1 we would have

$$x_1(i) \leftarrow \alpha_1 x_1(i-1) + \alpha_2 x_2(i-1) + \alpha_5 x_5(i-1) \ \ \text{(for node 1)}$$

where $x_1(i)$ denotes the updated measurement of node 1 at iteration i, and the $\alpha's$ are appropriately chosen coefficients. Every other node in the network performs the same operation. Under suitable conditions on the $\alpha's$, all node measurements will converge asymptotically to the desired average temperature $\bar{T}$. This consensus implementation requires only local communications. It also requires that each node performs several *iterations* before its updated measurement approaches the desired average temperature.

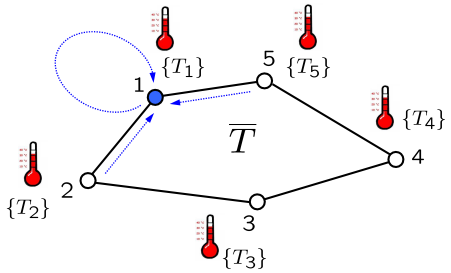

Fig. 1. A distributed network with N nodes accessing temperature data.

1.2. Incremental and Diffusion Strategies

Obviously, the effectiveness of any distributed implementation will depend on the modes of cooperation that are allowed among the nodes. Figure 2 illustrates three such modes of cooperation.

In an incremental mode of cooperation, information flows in a sequential manner from one node to the adjacent node. This mode of operation requires a cyclic pattern of collaboration among the nodes, and it tends to require the least amount of communications and power [5, 6, 7]. In a diffusion implementation, on the other hand, each node communicates with all its neighbors as dictated by the network topology. The amount of communication in this case is higher than in an incremental solution. Nevertheless, the nodes have access to more data from their neighbors. The communications in the diffusion implementation can be reduced by allowing each node to communicate only with a subset of its neighbors.

This material was based on work supported in part by the National Science Foundation under award ECS-0601266. The work of C. G. Lopes was also partially supported by a fellowship from CAPES, Brazil, under award 1168/01-0. This survey article is based on the works appearing in the publications [12]–[15].

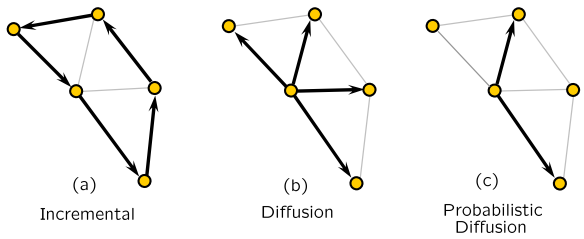

Fig. 2. Three modes of cooperation.

In this mode of cooperation the choice of which subset of neighbors to communicate with can be randomized according to some performance criterion, constituting a probabilistic diffusion implementation.

1.3. Distributed Strategies

The temperature example that we mentioned before is a special case of a strategy for distributed processing known as consensus (e.g., [8, 9, 10]). Broadly, consensus functions as follows. Assume the network is interested in estimating a certain parameter. Each node collects observations over a period of time and reaches an individual decision about the parameter. During this time, there is limited interaction among the nodes; the nodes act more like individual agents. Following this initial stage, the nodes combine their estimates through several consensus iterations and generally converge asymptotically close to the desired (global) estimate of the parameter – see Fig. 3.

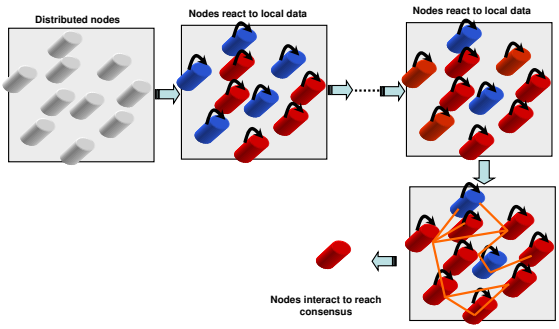

Fig. 3. An illustration of a consensus implementation.

Let us consider another example of a consensus implementation, which will serve as further motivation for the contributions in this presentation. Consider again a collection of nodes. Each node has access to a data vector y_k and a data matrix H_k. The y_k are noisy and distorted measurements of some unknown vector w^o:

$$y_k = H_k w^o + v_k$$

Each node can evaluate the least-squares estimator of w^o based on its own local data $\{y_k, H_k\}$. To do so, each node evaluates its local cross-correlation vector $\theta_k = H_k^* y_k$ and its auto-correlation matrix $R_k = H_k^* H_k$. Then the local estimate of w^o can be found from $\hat{w}_k = R_k^{-1}\theta_k$. This operation requires that each node collects sufficient data into y_k and H_k. Once the local quantities $\{\theta_k, R_k\}$ have been evaluated at the individual nodes, one can apply consensus iterations at the nodes to converge to the average quantities [11]:

$$R = \frac{1}{N}\sum_{k=1}^{N} R_k \qquad \text{and} \qquad \theta = \frac{1}{N}\sum_{k=1}^{N} \theta_k$$

The global estimate of w^o is given by $\hat{w} = \hat{R}^{-1}\hat{\theta}$.

For all practical purposes, a least-squares implementation in this manner is a non-recursive (or non-adaptive) solution. For example, if a node collects one more entry in y_k and one more row in H_k, the consensus iterations will need to be repeated afresh instead of being updated. In addition, the block averaging procedure limits the ability of the consensus-based solution to track fast-changing environments, especially in networks with limited communication resources.

1.4. Adaptive Networks

Motivated by these observations, we review in this article distributed algorithms that enable a network of nodes to function as an adaptive entity in its own right following the works [12]–[15]. In order to clarify what we mean by an adaptive network, let us first review the structure of a traditional adaptive filter. As is well known, and as shown in Fig. 4, an adaptive filter is generally a digital filter that changes its internal structure in response to an excitation and a reference signal. At each time instant, the filter compares its output to a reference signal and generates an error signal. The filter then adjusts its coefficients depending on whether the error is large or small. Thus, the key fact to note is that a regular

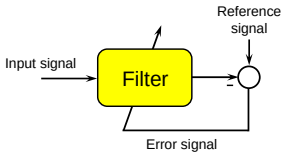

Fig. 4. An adaptive filter structure.

adaptive filter responds in real-time to its data and to variations in the statistical properties of this data. We want to extend this ability to the network domain. By an adaptive network we mean an interconnected structure of adaptive nodes that is able to respond to data in real-time and to track variations in the statistical properties of the data as well. As a result, in an adaptive network, whenever information arrives at a particular node, the information creates a ripple effect throughout the network and it influences the performance and behavior of the other nodes as dictated by the network topology. Let us illustrate the concept of an adaptive network by reconsidering the earlier example of Fig. 3.

Consider again a collection of nodes and assume the network is required to estimate a certain parameter of interest – see Fig. 5. In an adaptive network, each node collects local observations and at the same time interacts with its immediate neighbors. At every instant, the local observation is combined with information from the neighboring nodes in order to improve the estimate at the local node. By repeating this process of simultaneous observation and consultation, the nodes are constantly exhibiting updated estimates that respond to observations in real time.

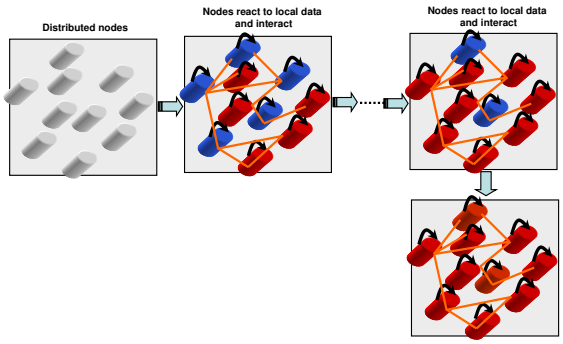

Fig. 5. An illustration of an adaptive network strategy.

1.5. Notation

We use boldface letters for random quantities and normal font for non-random (deterministic) quantities. We also use capital letters for matrices and small letters for vectors. For example, $\boldsymbol{d}$ is a random quantity and d is a realization or measurement for it, and R is a covariance matrix while w is a weight vector. The notation * denotes complex conjugation for scalars and complex-conjugate transposition for matrices.

2. INCREMENTAL LMS SOLUTION

Consider a network with N nodes (see Fig. 6). Each node k has access to time-realizations $\{d_k(i), u_{k,i}\}$ of zero-mean spatial data $\{\boldsymbol{d}_k, \boldsymbol{u}_k\}$, $k = 1, \ldots, N$, where each $\boldsymbol{d}_k$ is a scalar measurement and each $\boldsymbol{u}_k$ is a $1 \times M$ row regression vector. We collect the regression and measurement data into two global matrices:

$$\boldsymbol{U} \triangleq \mathrm{col}\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N\} \quad (N \times M) \tag{1}$$

$$\boldsymbol{d} \triangleq \mathrm{col}\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_N\} \quad (N \times 1) \tag{2}$$

These quantities collect the data across all N nodes. The objective is to estimate the $M \times 1$ vector w that solves

$$\min_{w} \; J(w) \tag{3}$$

where the cost function $J(w)$ denotes the mean-square error:

$$J(w) = E\|\boldsymbol{d} - \boldsymbol{U}w\|^2 \tag{4}$$

and E is the expectation operator. The optimal solution w^o of (3) satisfies the *normal equations* [16]:

$$R_{du} = R_u w^o \tag{5}$$

which are defined in terms of the correlation and cross-correlation quantities:

$$R_u = E\,\boldsymbol{U}^*\boldsymbol{U} \quad (M \times M), \qquad R_{du} = E\,\boldsymbol{U}^*\boldsymbol{d} \quad (M \times 1) \tag{6}$$

If the optimal solution w^o were to be computed from (5), then *every node* in the network would need to have access to the global statistical information $\{R_u, R_{du}\}$. Alternatively, the solution w^o could be computed centrally and the result broadcast to all nodes. Either way, these approaches drain considerable communications and computational resources and they do not endow the network with the necessary adaptivity to cope with possible changes in the statistical properties of the data. We shall instead describe *distributed* solutions that allow cooperation among the nodes through limited local communications, while at the same time equipping the network with adaptive mechanisms [12]–[15].

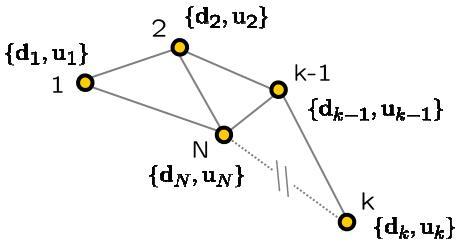

Fig. 6. A distributed network with N active nodes accessing space-time data.

2.1. Steepest-Descent Solution

To begin with, note from (4) and (6) that the cost function $J(w)$ can be decomposed as

$$J(w) = \sum_{k=1}^{N} J_k(w) \tag{7}$$

where each $J_k(w)$ is given by

$$\begin{aligned} J_k(w) &\triangleq E|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2 \\ &= \sigma_{d,k}^2 - R_{ud,k}w - w^* R_{du,k} + w^* R_{u,k} w \end{aligned} \tag{8}$$

and the second-order moment quantities are defined by

$$\sigma_{d,k}^2 = E|\boldsymbol{d}_k|^2, \quad R_{u,k} = E\,\boldsymbol{u}_k^*\boldsymbol{u}_k \quad \text{and} \quad R_{du,k} = E\,\boldsymbol{d}_k\boldsymbol{u}_k^* \tag{9}$$

In other words, $J(w)$ can be expressed as the sum of N individual cost functions $J_k(w)$, one for each node k. There have been extensive works in the literature on incremental methods for solving such optimization problems in a distributed manner (e.g., [5, 6, 17, 18]). Essentially, whenever a cost function can be decoupled into a sum of individual cost functions, a distributed algorithm can be developed for minimizing the cost function through an incremental procedure. We explain the procedure as follows in the context of mean-square-error estimation.

Thus recall that the traditional iterative steepest-descent solution for determining w^o can be expressed in the form:

$$w_i = w_{i-1} - \mu\,[\nabla J(w_{i-1})]^*\,, \tag{10}$$

$$= w_{i-1} + \mu \sum_{k=1}^{N} (R_{du,k} - R_{u,k} w_{i-1}) \tag{11}$$

where $\mu > 0$ is a suitably chosen positive step-size parameter, w_i is an estimate for w^o at iteration i, and $\nabla J(w_{i-1})$ denotes the gradient vector of $J(w)$ w.r.t. w evaluated at w_{i-1}. An equivalent implementation can be motivated as follows.

Let us define a *cycle* visiting every node over the network topology only once, such that each node has access only to its

immediate neighbor node in this cycle and let $\psi_k^{(i)}$ denote a *local estimate* of w^o at node k at time i. Thus assume that node k has access to $\psi_{k-1}^{(i)}$, which is an estimate of w^o at its immediate neighbor node $k-1$ in the defined cycle (see Fig. 7). If at each time instant i, we start with the initial condition $\psi_0^{(i)} = w_{i-1}$ at node 1 (i.e., with the current global estimate w_{i-1} for w^o), and iterate cyclicly across the nodes then, at the end of the procedure, the local estimate at node N will coincide with w_i from (10), i.e., $\psi_N^{(i)} = w_i$. In other words, the following implementation is equivalent to (10):

$$\left\{\begin{array}{l} \psi_0^{(i)} = w_{i-1} \\ \psi_k^{(i)} = \psi_{k-1}^{(i)} - \mu_k \left[\nabla J_k(w_{i-1})\right]^*, \quad k = 1, \ldots, N \\ w_i = \psi_N^{(i)} \end{array}\right. \tag{12}$$

observe that in this steepest-descent implementation, the iteration for $\psi_k^{(i)}$ is over the spatial index k.

2.2. Steepest-Descent Solution

Although recursion (12) is cooperative in nature, with each node k using information from its immediate neighbor (represented by $\psi_{k-1}^{(i)}$), this implementation still requires the nodes to have access to the global information w_{i-1} in order to evaluate $\nabla J_k(w_{i-1})$.

In order to resolve this difficulty and arrive at a distributed implementation, we rely on incremental techniques. If each node evaluates the required partial gradient $\nabla J_k(\cdot)$ at the local estimate $\psi_{k-1}^{(i)}$ received from node $k-1$, as opposed to w_{i-1}, then an incremental version of algorithm (12) would result, namely,

$$\left\{\begin{array}{l} \psi_0^{(i)} = w_{i-1} \\ \psi_k^{(i)} = \psi_{k-1}^{(i)} - \mu_k \left[\nabla J_k(\psi_{k-1}^{(i)})\right]^*, \quad k = 1, \ldots, N \\ w_i = \psi_N^{(i)} \end{array}\right. \tag{13}$$

This cooperative scheme relies only on locally available information, leading to a truly distributed solution. The scheme requires each node to communicate *only* with its immediate neighbor, thus saving on communication and energy resources [5, 6, 19].

2.3. Incremental Adaptive Solution

The incremental solution (13) relies on knowledge of the second-order moments $R_{du,k}$ and $R_{u,k}$, which are needed to evaluate the local gradients ∇J_k. An adaptive implementation of (13) can be obtained by replacing the second-order moments $\{R_{du,k}, R_{u,k}\}$ by instantaneous approximations, say of the LMS type, as follows

$$R_{du,k} \approx d_k(i) u_{k,i}^*, \quad R_{u,k} \approx u_{k,i}^* u_{k,i} \tag{14}$$

by using data realizations $\{d_k(i), u_{k,i}\}$ at time i. The approximations (14) lead to an adaptive distributed incremental algorithm, or simply a *distributed incremental LMS* algorithm of the form [12]–[15]:

For each time $i \geq 0$, repeat:

$$\left\{\begin{array}{l} \psi_0^{(i)} = w_{i-1} \\ \psi_k^{(i)} = \psi_{k-1}^{(i)} + \mu_k u_{k,i}^* \left(d_k(i) - u_{k,i}\psi_{k-1}^{(i)}\right) \\ \qquad k = 1, \ldots, N \\ w_i = \psi_N^{(i)} \end{array}\right. \tag{15}$$

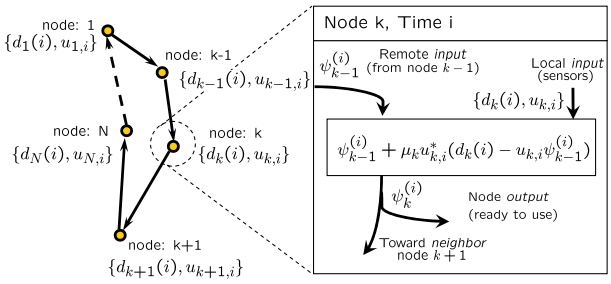

Fig. 7. Data processing in the incremental adaptive LMS solution.

3. INCREMENTAL LEAST-SQUARES SOLUTIONS

The use of LMS-type distributed algorithms eliminates the need to embed powerful processors at the nodes. However, as available processors continuously decrease in cost and increase in computational capability, one may consider equipping the network with more sophisticated adaptation rules. We now describe an incremental RLS implementation [14].

Again, each node k has access to regressor and measurement data $u_{k,i}$ and $d_k(i)$, $k = 1, \ldots, N$. At each time instant i, the entire network has access to space-time data

$$y_i = \left[\begin{array}{c} d_1(i) \\ d_2(i) \\ \vdots \\ d_N(i) \end{array}\right] \quad \text{and} \quad H_i = \left[\begin{array}{c} u_{1,i} \\ u_{2,i} \\ \vdots \\ u_{N,i} \end{array}\right]. \tag{16}$$

Here y_i and H_i are snapshot matrices unveiling the network data status at time i. We then formulate an exponentially weighted regularized least-squares (LS) problem [16], where the weight vector estimate w_i is found by solving:

$$\min_w \left[\lambda^{i+1} w^* \Pi w + \big(\mathcal{Y}_i - \mathcal{H}_i w\big)^* \mathcal{W}_i \big(\mathcal{Y}_i - \mathcal{H}_i w\big)\right] \tag{17}$$

and the weighting matrix is given by

$$\mathcal{W}_i = \text{diag}\{\lambda^i D, \lambda^{i-1} D, \cdots, \lambda D, D\} \tag{18}$$

with a spatial weighting matrix

$$D = \text{diag}\{\gamma_1, \gamma_2, \cdots, \gamma_N\} \tag{19}$$

and a (time) forgetting factor

$$0 \ll \lambda \leq 1. \tag{20}$$

Moreover, $\mathcal{Y}_i$ and $\mathcal{H}_i$ collect all the data blocks available from the beginning of the observation period up to current time

$$\mathcal{Y}_i = \left[\begin{array}{c} y_0 \\ \hline y_1 \\ \hline \vdots \\ \hline y_i \end{array}\right] \quad \text{and} \quad \mathcal{H}_i = \left[\begin{array}{c} H_0 \\ \hline H_1 \\ \hline \vdots \\ \hline H_i \end{array}\right]. \tag{21}$$

The global data matrices $\mathcal{Y}_i$ and $\mathcal{H}_i$ exhibit space-time structure, which naturally suggests a distributed solution. An algorithm that updates w_i recursively and in a distributed fashion is given by [14]:

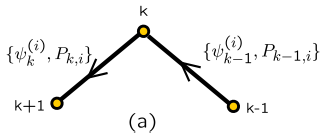

Fig. 8. The cooperation strategy of the exact distributed RLS algorithm (dRLS) described by (22).

$$
\begin{array}{l}
\psi_0^{(i)} \leftarrow w_{i-1}; \quad P_{0,i} \leftarrow \lambda^{-1} P_{i-1} \\
\text{for } k = 1 : N \\
\quad e_k(i) = d_k(i) - \boldsymbol{u}_{k,i} \psi_{k-1}^{(i)} \\
\quad \psi_k^{(i)} = \psi_{k-1}^{(i)} + \frac{P_{k-1,i}}{\gamma_k^{-1} + \boldsymbol{u}_{k,i} P_{k-1,i} \boldsymbol{u}_{k,i}^*} \boldsymbol{u}_{k,i}^* e_k(i) \\
\quad P_{k,i} = P_{k-1,i} - \frac{P_{k-1,i} \boldsymbol{u}_{k,i}^* \boldsymbol{u}_{k,i} P_{k-1,i}}{\gamma_k^{-1} + \boldsymbol{u}_{k,i} P_{k-1,i} \boldsymbol{u}_{k,i}^*} \\
\text{end} \\
w_i \leftarrow \psi_N^{(i)}; \quad P_i \leftarrow P_{N,i} \, .
\end{array}
\tag{22}
$$

Note that in algorithm (22) the iterations are performed over the spatial index k, therefore a path is induced across the network, along which w_{i-1} is spatially updated by sequentially visiting every node once. At each time i, the estimate $\psi_k^{(i)}$ at node k is the LS solution considering data blocks $\mathcal{Y}_{i-1}$ and $\mathcal{H}_{i-1}$ in addition to the data collected along the path. At the end of the cycle, $\psi_N^{(i)}$ will contain precisely the desired solution w_i. If we start from $i = 0$ with $w_{-1} = 0$ and $P_{-1} = \Pi^{-1}$ and repeatedly apply (22) taking into account sequentially all the data blocks up to time i, then by induction $\psi_N^{(i)}$ (or w_i) will be the solution to the *global* LS problem (17). Figure 8 depicts the structure of the incremental implementation, in which both $\psi_k^{(i)}$ and $P_{k,i}$ are transmitted to the next node in the path. This distributed implementation saves communication and energy resources in comparison to consensus-based strategies [11].

A simplification that requires less communications while keeping the performance close to the exact implementation can be obtained as follows. We allow collaboration for the estimates while keeping the matrices $P_{k,i}$ evolving locally and independent from the neighbor nodes. This would lead to the following algorithm [14]:

$$
\begin{array}{l}
\psi_0^{(i)} \leftarrow w_{i-1}; \quad P_{0,i} \leftarrow \lambda^{-1} P_{i-1} \\
\text{for } k = 1 : N \\
\quad e_k(i) = d_k(i) - \boldsymbol{u}_{k,i} \psi_{k-1}^{(i)} \\
\quad \psi_k^{(i)} = \psi_{k-1}^{(i)} + \frac{P_{k,i-1}}{\gamma_k^{-1} + \boldsymbol{u}_{k,i} P_{k,i-1} \boldsymbol{u}_{k,i}^*} \boldsymbol{u}_{k,i}^* e_k(i) \\
\quad P_{k,i} = P_{k,i-1} - \frac{P_{k,i-1} \boldsymbol{u}_{k,i}^* \boldsymbol{u}_{k,i} P_{k,i-1}}{\gamma_k^{-1} + \boldsymbol{u}_{k,i} P_{k,i-1} \boldsymbol{u}_{k,i}^*} \\
\text{end} \\
w_i \leftarrow \psi_N^{(i)}; \quad P_i \leftarrow P_{N,i} \, .
\end{array}
\tag{23}
$$

Algorithm (23) iterates the estimates $\psi_k^{(i)}$ over space, while $P_{k,i}$ is iterated over time with local data only. As a consequence it requires transmission complexity $O(M)$ as opposed to $O(M^2)$ for (22). Figure 9 presents the algorithm's collaboration strategy, in which estimates are shared along the path and matrices $P_{k,i}$ evolve locally.

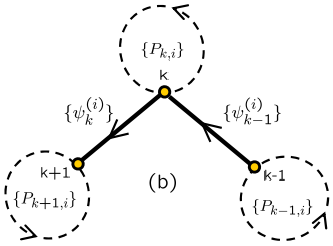

Fig. 9. The cooperation strategy of the low communications distributed RLS algorithm (LC-dRLS).

4. DIFFUSION LMS SOLUTION

When more communication resources are available, we may take advantage of the network connectivity and devise more sophisticated peer-to-peer cooperation rules. We describe one such diffusion protocol here [13, 15] – see Fig. 10. The neighborhood of a node k is the set of nodes directly connected to it, including itself. Each individual node k consults peer nodes from its neighborhood and combines their past estimates $\{\psi_\ell^{(i-1)}; \ell \in \mathcal{N}_k(i-1)\}$ with its own past estimate $\psi_k^{(i-1)}$. The node generates an aggregate estimate $\phi_k^{(i-1)}$ and feeds it in its local adaptive filter. The strategy can be expressed as follows for LMS-type recursions:

$$
\begin{aligned}
\phi_k^{(i-1)} &= f_k\left(\psi_\ell^{(i-1)}; \ell \in \mathcal{N}_k(i-1)\right) \\
\psi_k^{(i)} &= \phi_k^{(i-1)} + \mu u_{k,i}^* \left(d_k(i) - u_{k,i} \phi_k^{(i-1)}\right)
\end{aligned}
\tag{24}
$$

for some local combiner $f_k(\cdot)$. The combiners can be nonlinear or even time-variant, to reflect, for instance, changing topologies or to respond to non-stationary environments. One simple combining rule is to average the local and neighbors' previous estimates, i.e.,

$$
\begin{aligned}
\phi_k^{(i-1)} &= \sum_{\ell \in \mathcal{N}_k} a(k,\ell)\, \psi_\ell^{(i-1)} \\
\psi_k^{(i)} &= \phi_k^{(i-1)} + \mu u_{k,i}^* \left(d_k(i) - u_{k,i} \phi_k^{(i-1)}\right)
\end{aligned}
\tag{25}
$$

where $a(k,\ell) = 1/\deg(k)$, with $\deg(k)$ denoting the degree of node k (number of incident links at this node, including itself). This scheme exploits network connectivity more fully, leading to

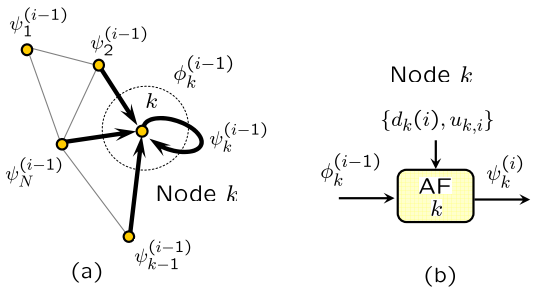

Fig. 10. A network with diffusion cooperation strategy.

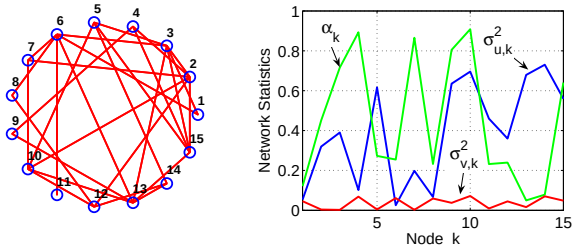

Fig. 11. Network Topology and Statistical profile.

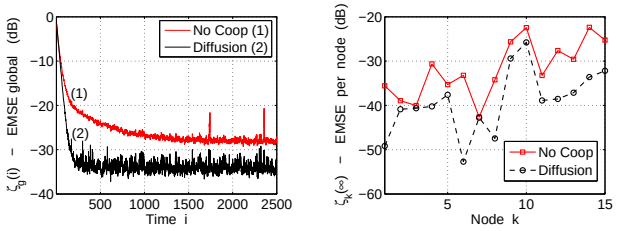

Fig. 12. Transient global EMSE and steady-state EMSE per node.

more robust algorithms. If links or nodes eventually fail, the adaptive network can react by relying on the remaining topology. Note that the adaptive network would work even for non-connected graphs, relying on the individual agents. Furthermore, since more information is aggregated in the local adaptive filter updates, individual nodes can attain better learning behavior when compared to the non-cooperative case, provided that the combiners f_k are well designed.

In order to illustrate the adaptive network performance, we present a simulation example in Figs. 11 and 12. Fig. 11 depicts the network topology and the network statistical profile. The regressors follow a first order Markov process with power $\sigma^2_{u,k}$ and correlation index α_k. The background noise power is denoted by $\sigma^2_{v,k}$. Note how the diffusion protocol outperforms the non-cooperative case. Fig. 12, left plot, presents the average global excess mean-square error (EMSE), defined as $\zeta_g(i) = \frac{1}{N}\sum_{k=1}^{N}\zeta_k(i)$, where the individual EMSE at node k is depicted in the right plot and is defined as $\zeta_k(i) = E|\boldsymbol{u}_{k,i}(w^o - \boldsymbol{\psi}_k^{(i-1)})|^2$.

5. CONCLUDING REMARKS AND FUTURE WORK

We have described several distributed and cooperative algorithms that endow distributed networks with learning abilities. They address distributed estimation problems that arise in a variety of applications, such as environment monitoring, target localization and potential sensor network problems [1].

For low energy profile implementations, the incremental LMS algorithm performs well. As the available resources increase, more sophisticated learning rules, such as recursive least-squares, can help speed network convergence. Still, with the increase in the size of networks, setting a cycle may not be a trivial task. In order to alleviate topology constraints and exploit more fully network connectivity, diffusion protocols can be developed. They give rise to peer-to-peer estimation protocols that exploit spatial diversity, improve robustness, and benefit the network in terms of estimation performance in comparison to the non-cooperative case. Diffusion protocols may also be extended to the RLS case.

An interesting extension of the diffusion protocol is to develop an extra layer over the existing adaptive network, in which the nodes are weighted according to their respective performance, instead of a blind aggregation of peer neighbors' estimates.

6. REFERENCES

[1] D. Estrin, G. Pottie and M. Srivastava, "Intrumenting the world with wireless sensor setworks," *Proc. ICASSP*, Salt Lake City, UT, May 2001, pp. 2033-2036.

[2] D. Li, K. D. Wong, Y. H. Hu and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, Issue 2, March 2002, pp. 17-29.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, Issue 8, August, 2002, pp. 102-114.

[4] D. Culler, D. Estrin and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, No. 8, Aug. 2004, pp. 41-49.

[5] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE Journal on Selected Areas in Communications*, vol.23, April 2005, No.4, pp. 798-808.

[6] D. Bertsekas, "A new class of incremental gradient methods for least squares problems," newblock *SIAM J. Optim.*, vol.7, No. 4, Nov. 1997, pp. 913-926.

[7] A. Nedic and D. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, No. 1, 2001, pp. 109-138.

[8] J. Tsitsiklis and M. Athans, "Convergence and asymptotic agreement in distributed decision problems," *IEEE Transactions on Automatic Control*, vol. AC-29, No. 1, Jan. 1984, pp. 42-50.

[9] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, Issue 9, Sept. 2004, pp. 1520-1533.

[10] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, Issue 1, Sep. 2004, pp. 65-78.

[11] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," *Fourth Internation Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, Apr. 2005, pp. 63-70.

[12] C. Lopes and A. H. Sayed, "Distributed adaptive incremental strategies: formulation and performance analysis," *Proc. ICASSP*, Toulouse, France, May 2006, vol. 3, pp. 584–587.

[13] C. G. Lopes and A. H. Sayed, "Distributed processing over adaptive networks," *Proc. Adaptive Sensor Array Processing Workshop*, MIT Lincoln Lab., Lexington, MA, June 2006.

[14] A. H. Sayed and C. Lopes, "Distributed recursive least-squares strategies over adaptive networks," *Proc. 40th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, October 2006.

[15] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *submitted for publication*.

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

[17] J. Tsitsiklis, D. P. Bertsekas and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. AC-31, No. 9, Sept. 1986, pp. 650-655.

[18] B. T. Poljak and Y. Z. Tsypkin, "Pseudogradient adaptation and training algorithms" *Automatic and Remote Control*, vol. 12, 1973, pp. 83-94.

[19] M. G. Rabbat and R. D. Nowak, "Decentralized source localization and tracking," *Proc. ICASSP*, Montreal, Canada, May 2004, vol. III, pp. 921-924.