Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 9831378, 8 pages
https://doi.org/10.1155/2018/9831378

Research Article

Memory Distributed LMS for Wireless Sensor Networks

Fangmin Xu, Chenyang Zheng, and Haiyan Cao

The School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, Zhejiang, China

Correspondence should be addressed to Fangmin Xu; xufangmin@hdu.edu.cn

Received 5 September 2017; Revised 5 February 2018; Accepted 7 February 2018; Published 4 March 2018

Academic Editor: Raquel Caballero-Águila

Due to the limited communication resource and power, it is usually infeasible for sensor networks to gather data to a central processing node. Distributed algorithms are an efficient way to resolve this problem. In the algorithms, each sensor node deals with its own input data and transmits the local results to its neighbors. Each node fuses the information from neighbors and its own to get the final results. Different from the existing work, in this paper, we present an approach for distributed parameter estimation in wireless sensor networks based on the use of memory. The proposed approach consists of modifying the cost function by adding extra statistical information. A distributed least-mean squares (d-LMS) algorithm, called memory d-LMS, is then derived based on the cost function and analyzed. The theoretical performances of mean and mean square are analyzed. Moreover, simulation results show that the proposed algorithm outperforms the traditional d-LMS algorithm in terms of convergence rate and mean square error (MSE) performance.

1. Introduction

Wireless Sensor Networks (WSNs) consist of a large number of sensor nodes that can be deployed for monitoring unattainable areas, such as deep oceans, forest fires, and air pollution [1–4]. To estimate some parameter of interest from data which is collected at nodes distributed over a geographic area, it is becoming more and more important to design and analyze estimation algorithms for the networks.

Generally, there are two kinds of estimation algorithms: centralized estimation [5–8] and distributed estimation [9–12]. In the former one, all nodes transmit their measurements to a central fusion center for processing, and the final estimate is sent back to the nodes. This method can obtain the global solution, but it requires a large amount of energy and communication. What is more, it will be nonrobust if the fusion center is out of work. In the latter one, each node only communicates with its immediate neighbor and the signal is processed locally. Compared to the centralized estimation, the distributed estimation can achieve good estimation performance whilst reducing complexity.

Due to the scalability, robustness, and low cost, distributed estimation algorithms have been receiving more and more attention. The paper [10] proposes a diffusion LMS to obtain the distributed estimation of the networks. The paper [11] proposes a distributed incremental RLS solution. The papers [12, 13] propose LMS algorithms for censor data. To the authors' knowledge, there is no existing work dealing with the past information of the nodes for distributed estimation. Motivated by this fact, in this paper, we state a new cost function where the past information is considered in the cost function. By optimizing the proposed cost function, a diffusion memory-LMS is proposed in this paper.

This paper is organized as follows. Section 2 describes the mathematical problem of the network and memory d-LMS. In Section 3, mean convergence is analyzed. Simulation results are provided in Section 4. Finally, we conclude the paper in Section 5.

Notation. We use boldface uppercase letters to denote matrices and boldface lowercase letters to denote vectors. We use $E(\cdot)$ to denote the expectation operator, $D(\cdot)$ for variance, $(\cdot)^T$ for complex conjugate-transposition, $\mathrm{Tr}(\cdot)$ for the trace of a matrix, and $\otimes$ for Kronecker product. The notation $\mathrm{col}\{\cdot\}$ stands for a vector obtained by stacking the specified vectors. $\mathrm{vec}(\cdot)$ stands for a linear transformation which converts a matrix into a column vector. Similarly, we use $\mathrm{diag}\{\cdot\}$ to denote the diagonal matrix consisting of the specified vectors or matrices. Other notations will be introduced as necessary.

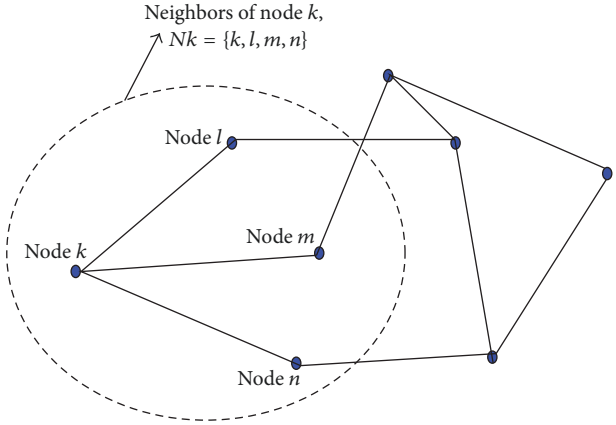

Figure 1: Wireless sensor networks with each node taking a measurement at time i and storing the measurements from time $i - i_0 + 1$ to time i.

2. The Mathematical Problem of the Network and Memory d-LMS

We consider a sensor network which is composed of K sensor nodes distributed over some geographic regions and used to monitor a physical phenomenon, shown in Figure 1. At every time i, each node k can get a scalar measurement $y_{k,i}$. Here $y_{k,i}$ is a 1×1 observations vector, and it can be expressed in terms of the following linear combinations:

$$y_{k,i} = x_{k,i}^T \omega_0 + e_{k,i}, \tag{1}$$

where $x_{k,i}$ is an $M \times 1$ known regression vector which is corresponding to a realization of a stochastic process $x_{k,i}$. ω_0 is an $M \times 1$ nonrandom unknown parameter to be estimated. $e_{k,i}$ is a random error in observing $y_{k,i}$ and is assumed to be spatially and temporally independent. $E(e_{k,i}) = 0$ and $D(e_{k,i}) = \sigma_{k,i}^2$. To get a global solution of ω_0, it requires access to the information $\{y_{k,i}, x_{k,i}\}$ across all nodes in the network. Therefore, we explore a distributed algorithm in this section.

2.1. Traditional Cost Function and d-LMS. Generally, the d-LMS algorithms [10–12] are derived from the following problem function:

$$J^{\text{global}}(\omega) \triangleq \sum_{k=1}^{K} \sum_{l \in N_k} c_{l,k} E \left| y_{l,i} - x_{l,i}^T \omega \right|^2 = \sum_{k=1}^{K} J_k^{\text{local}}(\omega), \tag{2}$$

where $\triangleq$ is used to denote $J_k^{\text{local}}(\omega)$ as $\sum_{l \in N_k} c_{l,k} E|y_{l,i} - x_{l,i}^T \omega|^2$, $J_k^{\text{local}}(\omega) \triangleq \sum_{l \in N_k} c_{l,k} E|y_{l,i} - x_{l,i}^T \omega|^2$ is the local cost function of node k. $\{c_{l,k}\}$ is a set of nonnegative weight coefficients between node l and node k and satisfies

$$\sum_{l=1}^{K} c_{l,k} = \sum_{k=1}^{K} c_{l,k} = 1. \tag{3}$$

Here, $C = \{c_{l,k}\}_{K*K}$, and $c_{l,k} = 0$ when node l and node k are not connected.

By minimizing (2), one can obtain the Adapt-then-Combine (ATC) diffusion LMS as follows [10]:

$$\begin{aligned} \varphi_{k,i} &= \omega_{k,i-1} + \mu_k \sum_{l \in N_k} c_{l,k} x_{l,i} \left(y_{l,i} - x_{l,i}^T \omega_{k,i-1} \right) \\ \omega_{k,i} &= \sum_{l \in N_k} a_{l,k} \varphi_{l,i}, \end{aligned} \tag{4}$$

where μ_k is the step size and $\{a_{l,k}\}$ is a set of nonnegative weight coefficients between node l and node k and satisfies

$$a_{l,k} = 0, \quad \text{if } l \notin N_k, \ 1^T A = 1^T. \tag{5}$$

Here, A is an $N \times N$ matrix with individual entries $\{a_{l,k}\}$.

The Adapt-then-Combine (ATC) d-LMS can be described as follows. Firstly, at time i, every node k obtains its local estimation $\varphi_{k,i}$ by exchanging the measurements $\{y_{l,i}, x_{l,i}\}$ across all its neighbor nodes. Then, all nodes combine the estimation from the first step with their neighbors. The traditional d-LMS algorithms obtain their estimation at time i by considering the measurements across its neighbor nodes only at time i. Rather than relying solely on current data and on the data shared with the neighbors at a particular time instant [9–12], we propose a distributed least-mean squares (d-LMS) algorithm called memory d-LMS in the following subsection.

2.2. Cost Function and Memory d-LMS with the Past Measurement Information. To use the past measurements more sufficiently, we first propose cost functions with the past information. At time i, the global cost function with the past information is given by

$$\begin{aligned} f^{\text{global}}(\omega) &\triangleq \sum_{k=1}^{K} \sum_{j=i-i_0+1}^{i} E \left| y_{l,j} - x_{l,j}^T \omega \right|^2 \\ &= \sum_{k=1}^{K} \sum_{l \in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E \left| y_{l,j} - x_{l,j}^T \omega \right|^2 \\ &= \sum_{k=1}^{K} f_k^{\text{local}}(\omega), \end{aligned} \tag{6}$$

where

$$f_k^{\text{local}}(\omega) = \sum_{l \in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E \left| y_{l,j} - x_{l,j}^T \omega \right|^2 \tag{7}$$

is the local cost function with the past information. $\{c_{l,k}\}$ is a set of nonnegative weight coefficients between node l and node k and satisfies

$$\sum_{l=1}^{K} c_{l,k} = \sum_{k=1}^{K} c_{l,k} = 1. \tag{8}$$

Here, $C = \{c_{l,k}\}_{K \times K}$, $c_{l,k} = 0$ when node l and node k are not connected. i_0 ($1 \le i_0 \le i$) is a constant to be designed by requirements of the network. (6) is simplified into (2) if $i_0 = 1$.

By directly minimizing $f^{\text{global}}(\omega)$ in (6), we obtain a *global d-LMS* recursion

$$\begin{aligned}\omega_i &= \omega_{i-1} + \mu \sum_{k=1}^{K} \sum_{j=i-i_0+1}^{i} \left(R_{yx,k,j} - R_{xx,k,j}\omega_{i-1}\right) \\ &\approx \omega_{i-1} + \mu \sum_{k=1}^{K} \sum_{j=i-i_0+1}^{i} \left(y_{k,j}x_{k,j} - x_{k,j}x_{k,j}^T\omega_{i-1}\right),\end{aligned} \tag{9}$$

where $R_{yx,l,j} = E(x_{l,j}y_{l,j})$, $R_{xx,l,j} = E(x_{l,j}x_{l,j}^T)$, and μ is the step size. The moments $R_{yx,l,j}$ and $R_{xx,l,j}$ are usually unavailable in practice. Thus, we replace them by their instantaneous approximations $y_{k,j}x_{k,j}$ and $x_{k,j}x_{k,j}^T$, respectively [13].

Equation (9) is not distributed because it requires access to measurements across the whole network.

Now, we focus on a distributed solution of ω_0 by minimizing a local cost function with the past information. Notice that (6) can be rewritten as

$$f^{\text{global}}(\omega) = f_k^{\text{local}}(\omega) + \sum_{\substack{l=1,\\ l\neq k}}^{K} f_l^{\text{local}}(\omega), \tag{10}$$

where $f_k^{\text{local}}(\omega)$ can be rewritten as

$$\begin{aligned} f_k^{\text{local}}(\omega) &= \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E\left|y_{l,j} - x_{l,j}^T\omega\right|^2 \\ &= \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E\left|y_{l,j} - x_{l,j}^T\omega_k^{\text{local}} + x_{l,j}^T\omega_k^{\text{local}} - x_{l,j}^T\omega\right|^2 \\ &= \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E\left|y_{l,j} - x_{l,j}^T\omega_k^{\text{local}}\right|^2 \\ &\quad + \left(\omega - \omega_k^{\text{local}}\right)^T \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} Ex_{l,j}x_{l,j}^T \left(\omega - \omega_k^{\text{local}}\right) \\ &= \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E\left|y_{l,j} - x_{l,j}^T\omega_k^{\text{local}}\right|^2 \\ &\quad + \left\|\left(\omega - \omega_k^{\text{local}}\right)\right\|_{\Lambda_k}^2, \end{aligned} \tag{11}$$

where $\|(\omega - \omega_k^{\text{local}})\|_{\Lambda_k}^2 = (\omega - \omega_k^{\text{local}})^T\Lambda_k(\omega - \omega_k^{\text{local}})$ is a weighted vector norm and $\Lambda_k = \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} Ex_{l,j}x_{l,j}^T$, ω_k^{local} is the local estimate of node k.

Substituting (11) into (10) leads to

$$\begin{aligned} f^{\text{global}}(\omega) &= \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E\left|y_{l,j} - x_{l,j}^T\omega\right|^2 \\ &\quad + \sum_{l\neq k}^{K} \sum_{l'\in N_l} c_{l',l} \sum_{j=i-i_0+1}^{i} E\left|y_{l',j} - x_{l',j}^T\omega_l^{\text{local}}\right|^2 \\ &\quad + \sum_{l\neq k}^{K} \left\|\left(\omega - \omega_l^{\text{local}}\right)\right\|_{\Lambda_l}^2. \end{aligned} \tag{12}$$

Minimizing (12) with respect to ω, we have

$$\begin{aligned} \frac{df^{\text{global}}(\omega)}{d\omega} &= 2\sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E\left(x_{l,j}y_{l,j} - x_{l,j}x_{l,j}^T\omega\right) \\ &\quad + 2\sum_{l\neq k}^{K} \Lambda_l\left(\omega_l^{\text{local}} - \omega\right). \end{aligned} \tag{13}$$

Notice that estimation of ω_0 based on (13) still requires access to information across all nodes in the network. In response to this, we replace Λ_l by another kind of weighting coefficient $\gamma_{l,k}$, where

$$\gamma_{l,k} = 0, \quad \text{if } l \notin N_k. \tag{14}$$

Then, we have

$$\begin{aligned} \frac{df^{\text{global}}(\omega)}{d\omega} &\approx 2\sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} E\left(x_{l,j}y_{l,j} - x_{l,j}x_{l,j}^T\omega\right) \\ &\quad + 2\sum_{l\neq k}^{K} \gamma_{l,k}\left(\omega_l^{\text{local}} - \omega\right) \\ &= 2\sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} \left(R_{yx,l,j} - R_{xx,l,j}\omega\right) \\ &\quad + 2\sum_{l\in N_k/\{k\}} \gamma_{l,k}\left(\omega_l^{\text{local}} - \omega\right). \end{aligned} \tag{15}$$

Hence, we get an iterative steepest-descent estimation for ω_0: that is,

$$\begin{aligned} \omega_{k,i} &= \omega_{k,i-1} + \mu_k \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} \left(R_{yx,l,j} - R_{xx,l,j}\omega_{k,i-1}\right) \\ &\quad + \eta_k \sum_{l\in N_k/\{k\}} \gamma_{l,k}\left(\omega_l^{\text{local}} - \omega_{k,i-1}\right), \end{aligned} \tag{16}$$

where μ_k and η_k are step size parameters.

We replace ω_l^{local} by $\omega_{l,i-1}$ which is available at node l in time $i-1$. Then, (15) is rewritten as

$$\begin{aligned} \omega_{k,i} &= \omega_{k,i-1} + \mu_k \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} \left(R_{yx,l,j} - R_{xx,l,j}\omega_{k,i-1}\right) \\ &\quad + \eta_k \sum_{l\in N_k/\{k\}} \gamma_{l,k}\left(\omega_{k,i-1} - \omega_{l,i-1}\right). \end{aligned} \tag{17}$$

Now, we use a two-step estimation for ω_0

$$\varphi_{k,i} = \omega_{k,i-1} + \mu_k \sum_{l\in N_k} c_{l,k} \sum_{j=i-i_0+1}^{i} \left(R_{yx,l,j} - R_{xx,l,j}\omega_{k,i-1}\right), \tag{18}$$

$$\begin{aligned}\omega_{k,i} &= \varphi_{k,i} + \eta_k \sum_{l\neq k, l\in N_k} \gamma_{l,k}\left(\omega_{l,i-1} - \omega_{k,i-1}\right)\\ &= \varphi_{k,i} + \eta_k \sum_{l\in N_k/\{k\}} \gamma_{l,k}\left(\varphi_{l,i} - \varphi_{k,i}\right)\\ &= \varphi_{k,i} - \eta_k \sum_{l\in N_k} \gamma_{l,k}\varphi_{k,i} + \eta_k\gamma_{k,k}\varphi_{k,i}\\ &\quad + \eta_k \sum_{l\in N_k/\{k\}} \gamma_{l,k}\varphi_{l,i}\\ &= \left(1 - \eta_k \sum_{l\in N_k} \gamma_{l,k} + \eta_k\gamma_{k,k}\right)\varphi_{k,i} + \eta_k \sum_{l\in N_k/\{k\}} \gamma_{l,k}\varphi_{l,i}\\ &\triangleq a_{k,k}\varphi_{k,i} + \sum_{l\in N_k/\{k\}} a_{l,k}\varphi_{l,i},\end{aligned} \tag{19}$$

where the last step of (19) follows from a change of variables

$$\begin{aligned}a_{k,k} &= 1 - \eta_k \sum_{l\in N_k} \gamma_{l,k} + \eta_k\gamma_{k,k},\\ a_{l,k} &= \eta_k\gamma_{l,k},\\ &\qquad l \neq k.\end{aligned} \tag{20}$$

Form (13) and (20), we have $a_{l,k} = \eta_k\gamma_{l,k} = 0$ if $l \notin N_k$.

Furthermore, from (20), we have $\sum_{l=1}^{N} a_{l,k} = 1$ directly.

To obtain an adaptive implementation, we replace the second-order moments by instantaneous values in (18): that is,

$$R_{yx,l,j} \approx y_{l,j}x_{l,j} \tag{21}$$

$$R_{xx,l,j} \approx x_{l,j}x_{l,j}^T. \tag{22}$$

Now, we obtain a novel distributed algorithm called *memory d-LMS*:

$$\varphi_{k,i} = \omega_{k,i-1} + \mu_k \sum_{l\in N_k} c_{l,k}\left(\alpha_{l,i} - \beta_{l,i}\omega_{k,i-1}\right) \tag{23}$$

$$\omega_{k,i} = \sum_{l\in N_k} a_{l,k}\varphi_{l,i}, \tag{24}$$

where $a_{l,k} = c_{l,k} = 0$, if $l \notin N_k$.

$$\begin{aligned}&\sum_{l=1}^{K} c_{l,k} = \sum_{k=1}^{K} c_{l,k} = 1,\\ &\sum_{l=1}^{K} a_{l,k} = 1,\\ &\alpha_{l,i} = \sum_{j=i-i_0+1}^{i} y_{l,j}x_{l,j};\\ &\beta_{l,i} = \sum_{j=i-i_0+1}^{i} x_{l,j}x_{l,j}^T.\end{aligned} \tag{25}$$

2.3. Implementation of the Proposed Memory d-LMS Algorithm. According to the abovementioned memory d-LMS, we implement the algorithm with three steps: time diffusion, adaptation, and node diffusion. In detail, it will be described as follows.

For every node k, start with $\{\omega_{k,0}\} = 0$, select an appropriate value for the step size μ_k, and set the nonnegative coefficients $a_{l,k}$, $c_{l,k}$ satisfying (24), for each time i, and repeat.

Step 1. Time diffusion step with respect to past measurements:

$$\begin{aligned}\alpha_{l,i} &= \sum_{j=i-i_0+1}^{i} y_{l,j}x_{l,j};\\ \beta_{l,i} &= \sum_{j=i-i_0+1}^{i} x_{l,j}x_{l,j}^T.\end{aligned} \tag{26}$$

In this step, at the ith iteration, every node l summarizes cross-correlations $y_{l,j}x_{l,j}$ from time $i - i_0 + 1$ to time i and autocorrelations $x_{l,j}x_{l,j}^T$ from time $i - i_0 + 1$ to time i, respectively.

Step 2. Adaptation step is

$$\varphi_{k,i} = \omega_{k,i-1} + \mu_k \sum_{l\in N_k} c_{l,k}\left(\alpha_{l,i} - \beta_{l,i}\omega_{k,i-1}\right). \tag{27}$$

Step 3. Node diffusion step with respect to preestimations of all nodes

$$\omega_{k,i} = \sum_{l\in N_k} a_{l,k}\varphi_{l,i}. \tag{28}$$

In this step, every node k combines its preestimation at Step 2 with preestimations shared by the neighbor nodes to obtain estimation $\omega_{k,i}$.

3. Performance Analysis

In this section, we analyze the asymptotic performance of the proposed algorithm.

Denote that

$$\begin{aligned}&\widetilde{\varphi}_{k,i} = \omega_0 - \varphi_{k,i}, \quad \forall k, i\\ &\widetilde{\omega}_{k,i} = \omega_0 - \omega_{k,i}\\ &\widetilde{\boldsymbol{\omega}}_i = \operatorname{col}\left\{\widetilde{\omega}_{1,i}, \widetilde{\omega}_{2,i}, \ldots, \widetilde{\omega}_{K,i}\right\}\\ &\widetilde{\boldsymbol{\omega}}_{i-1} = \operatorname{col}\left\{\widetilde{\omega}_{1,i-1}, \widetilde{\omega}_{2,i-1}, \ldots, \widetilde{\omega}_{K,i-1}\right\}\\ &\widetilde{\boldsymbol{\varphi}}_i = \operatorname{col}\left\{\widetilde{\varphi}_{1,i}, \widetilde{\varphi}_{2,i}, \ldots, \widetilde{\varphi}_{K,i}\right\}\\ &\mathbf{V}_i = \operatorname{col}\left\{\sum_{j=i-i_0+1}^{i} x_{1,j}e_{1,j}, \ldots, \sum_{j=i-i_0+1}^{i} x_{K,j}e_{K,j}\right\}\\ &\mathbf{M} = \operatorname{diag}\left\{\mu_1 I_M, \mu_2 I_M, \ldots, \mu_K I_M\right\}\end{aligned}$$

$$\mathbf{\Omega}_i = \operatorname{diag}\left\{\sum_{l\in N_1} c_{l,1}\sum_{j=i-i_0+1}^{i} x_{l,j}x_{l,j}^T,\ldots,\sum_{l\in N_K} c_{l,K}\cdot\sum_{j=i-i_0+1}^{i} x_{l,j}x_{l,j}^T\right\},$$

$$\mathbf{A} = A\otimes I_M = \begin{pmatrix} a_{11}I_M & \cdots & a_{1K}I_M \\ \vdots & \ddots & \vdots \\ a_{K1}I_M & \cdots & a_{KK}I_M \end{pmatrix}, \quad \text{where } A = \{a_{l,k}\}_{K*K}$$

$$\mathbf{C} = C\otimes I_M = \begin{pmatrix} c_{11}I_M & \cdots & c_{1K}I_M \\ \vdots & \ddots & \vdots \\ c_{K1}I_M & \cdots & c_{KK}I_M \end{pmatrix}, \quad \text{where } C = \{c_{l,k}\}_{K*K}. \tag{29}$$

Then, from (22) and (23), we have

$$\begin{aligned}\widetilde{\varphi}_{k,i} &= \widetilde{\omega}_{k,i-1} - \mu_k\sum_{l\in N_k} c_{l,k}\sum_{j=i-i_0+1}^{i} x_{l,j}x_{l,j}^T\widetilde{\omega}_{l,i-1} \\ &\quad - \mu_k\sum_{l\in N_k} c_{l,k}\sum_{j=i-i_0+1}^{i} x_{l,j}e_{l,j}, \\ \widetilde{\omega}_{k,i} &= \sum_{l\in N_k} a_{l,k}\widetilde{\varphi}_{l,i}.\end{aligned} \tag{30}$$

Furthermore, the above expressions can be rewritten as

$$\widetilde{\boldsymbol{\varphi}}_i = \widetilde{\boldsymbol{\omega}}_{i-1} - \mathbf{M}_i\mathbf{\Omega}_i\widetilde{\boldsymbol{\omega}}_{i-1} - \mathbf{C}^T\mathbf{M}_i\mathbf{V}_i, \tag{31}$$

$$\widetilde{\boldsymbol{\omega}}_i = \mathbf{A}^T\widetilde{\boldsymbol{\varphi}}_i. \tag{32}$$

Inserting (31) into (32), we have

$$\widetilde{\boldsymbol{\omega}}_i = \mathbf{A}^T\left[I - \mathbf{M}\mathbf{\Omega}_i\right]\widetilde{\boldsymbol{\omega}}_{i-1} - \mathbf{A}^T\mathbf{C}^T\mathbf{M}\mathbf{V}_i. \tag{33}$$

In the following of this section, we will give the asymptotic performance of $\widetilde{\boldsymbol{\omega}}_i$ and prove it under some assumptions.

3.1. Convergence Analysis of $E\omega_{k,i}$

Assumption 1. All regressors $\{x_{k,i}\}$ are independent of $\{e_{k,i}\}$ for all node k and time i.

Lemma 2 (see [10]). *Let P and Xdenote arbitrary $K\times K$ matrices, where $\mathbf{1}^TP = \mathbf{1}^T$. Then the matrix P^TX is stable for any choice of P if and only if X is stable.*

Theorem 3. *Under Assumption 1, mean of the memory d-LMS estimation based on the past information converges to the true value ω_0, if*

$$0 < i_0\mu_k < \frac{2}{\lambda_{\max}(\Sigma_k)}, \quad k = 1,\ldots,K, \tag{34}$$

where $\Sigma_k = (1/i_0)\sum_{j=i-i_0+1}^{i} x_{k,j}x_{k,j}^T$.

Proof. Taking the expectation of both sides of (34) yields

$$\begin{aligned}E\widetilde{\boldsymbol{\omega}}_i &= \mathbf{A}^T\left[I - \mathbf{M}E\mathbf{\Omega}_i\right]E\widetilde{\boldsymbol{\omega}}_{i-1} \\ &= \mathbf{A}^T\left[I - \mathbf{M}\cdot i_0\cdot\operatorname{diag}\{\Sigma_1,\ldots,\Sigma_K\}\right]E\widetilde{\boldsymbol{\omega}}_{i-1} \\ &= \left(\mathbf{A}^T\left[I - \mathbf{M}\cdot i_0\cdot\operatorname{diag}\{\Sigma_1,\ldots,\Sigma_K\}\right]\right)^i\omega_0\end{aligned} \tag{35}$$

The right side of (35) converges to 0, if $\mathbf{A}^T[I - \mathbf{M}\cdot i_0\cdot\operatorname{diag}\{\Sigma_1,\ldots,\Sigma_K\}]$ is stable. This requires $I - \mathbf{M}\cdot i_0\cdot\operatorname{diag}\{\Sigma_1,\ldots,\Sigma_K\}$ to be stable or equivalently

$$0 < i_0\cdot\mu_k < \frac{2}{\lambda_{\max}(\Sigma_k)}, \quad k = 1,\ldots,K. \tag{36}$$

Theorem 3 shows that the mean of the estimation convergences to the true value when proper μ_k is chosen. □

3.2. The Mean Squared Errors Performance.

Now, we study the mean squared errors of the memory d-LMS.

$$\begin{aligned}&E\left\|\widetilde{\boldsymbol{\omega}}_i\right\|_\Gamma^2 \\ &= E\left\|\widetilde{\boldsymbol{\omega}}_{i-1}\right\|^2_{[I-\mathbf{M}\mathbf{\Omega}_i]\mathbf{A}\Gamma\mathbf{A}^T[I-\mathbf{M}\mathbf{\Omega}_i]} \\ &\quad + E\left(\mathbf{V}_i^T\mathbf{M}\mathbf{C}\mathbf{A}\Gamma\mathbf{A}^T\mathbf{C}^T\mathbf{M}\mathbf{V}_i\right) \\ &\overset{(a)}{=} E\left\|\widetilde{\boldsymbol{\omega}}_{i-1}\right\|^2_{[I-\mathbf{M}\mathbf{\Omega}_i]\mathbf{A}\Gamma\mathbf{A}^T[I-\mathbf{M}\mathbf{\Omega}_i]} \\ &\quad + \operatorname{Tr}\left(\Gamma\mathbf{A}^T\mathbf{C}^TE\left(\mathbf{M}\mathbf{V}_i\mathbf{V}_i^T\mathbf{M}\right)\mathbf{C}\mathbf{A}\right) \\ &\overset{(b)}{=} E\left\|\widetilde{\boldsymbol{\omega}}_{i-1}\right\|^2_{[I-\mathbf{M}\mathbf{\Omega}_i]\mathbf{A}\Gamma\mathbf{A}^T[I-\mathbf{M}\mathbf{\Omega}_i]} \\ &\quad + \operatorname{vec}\left(\mathbf{A}^T\mathbf{C}^TE\left(\mathbf{M}\mathbf{V}_i\mathbf{V}_i^T\mathbf{M}\right)\mathbf{C}\mathbf{A}\right)^T\operatorname{vec}(\Gamma).\end{aligned} \tag{37}$$

Let $\tau = \operatorname{vec}(\Gamma)$ and $\Delta = \operatorname{vec}(\mathbf{A}^T\mathbf{C}^TE(\mathbf{M}\mathbf{V}_i\mathbf{V}_i^T\mathbf{M})\mathbf{C}\mathbf{A})^T$ in (37), by using the property

$$\operatorname{vec}(U\Gamma V) = \left(V^T\otimes U\right)\operatorname{vec}(\Gamma). \tag{38}$$

We rewrite (37) as

$$\begin{aligned}E\left\|\widetilde{\boldsymbol{\omega}}_i\right\|_\tau^2 &= E\left\|\widetilde{\boldsymbol{\omega}}_0\right\|^2_{F^i\tau} + \Delta\sum_{j=0}^{i-1}F^j\tau \\ &= E\left\|\widetilde{\boldsymbol{\omega}}_0\right\|^2_{F^i\tau} + \Delta\left(I - F^i\right)(I-F)^{-1}\tau,\end{aligned} \tag{39}$$

where

$$\begin{aligned}F &= E\left[\left(\left[I - \mathbf{M}\mathbf{\Omega}_i\right]\mathbf{A}\right)\otimes\left(\left[I - \mathbf{M}\mathbf{\Omega}_i\right]\mathbf{A}^T\right)\right] \\ &= E\left[\left(\left[I - \mathbf{M}\mathbf{\Omega}_i\right]\otimes\left[I - \mathbf{M}\mathbf{\Omega}_i\right]\right)\left(\mathbf{A}^T\otimes\mathbf{A}\right)\right].\end{aligned} \tag{40}$$

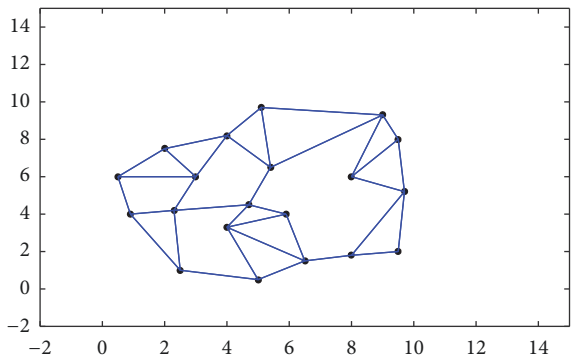

Figure 2: Network topology with $K = 20$ nodes.

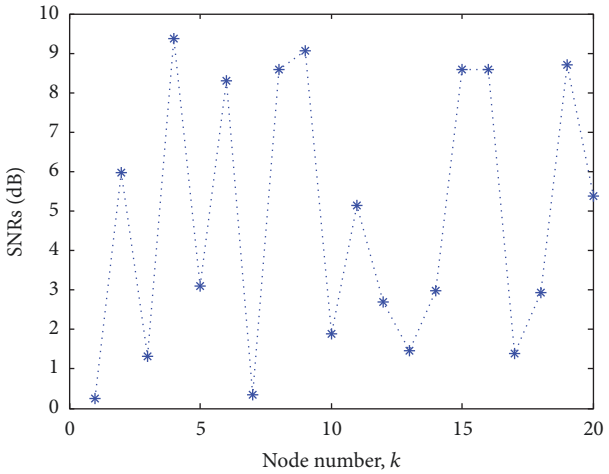

Figure 3: SNR levels varying across the nodes within $[0, 10]$ dB.

For sufficiently large i and stable matrix F, (39) can be rewritten as

$$E\left\|\widetilde{\boldsymbol{\omega}}_i\right\|_{\tau}^{2} = \Delta\left(I - F\right)^{-1}\tau. \tag{41}$$

Substituting $\tau = \mathrm{vec}(\mathrm{diag}(e_k) \otimes I_M)$ into (41), we can obtain the mean square deviation (MSD) as follows:

$$E\left\|\widetilde{\boldsymbol{\omega}}_{k,\infty}\right\|^{2} = \Delta\left(I - F\right)^{-1}\mathrm{vec}\left(\mathrm{diag}\left(e_k\right) \otimes I_M\right), \tag{42}$$

where e_k is a $K \times 1$ vector with a single entry 1 at position k and zeros elsewhere.

4. Simulations

In what follows, we present computer simulations to evaluate the performance of the proposed memory d-LMS algorithm and to verify the theoretical results introduced in Section 3.

Figure 2 gives the network topology with $K = 20$ nodes considered in this section. Besides, noise power levels, that is, $\sigma_{k,i}^2$, vary across the nodes within $[10^{-10}, 1]$. Correspondingly, SNRs vary within $[0, 10]$ dB which is shown in Figure 3, where SNR $= -\log_{10}(\sigma_{k,i}^2)$. The network data are generated based on model (1), and the unknown vector $\omega_0 = [-1, 3]^T$.

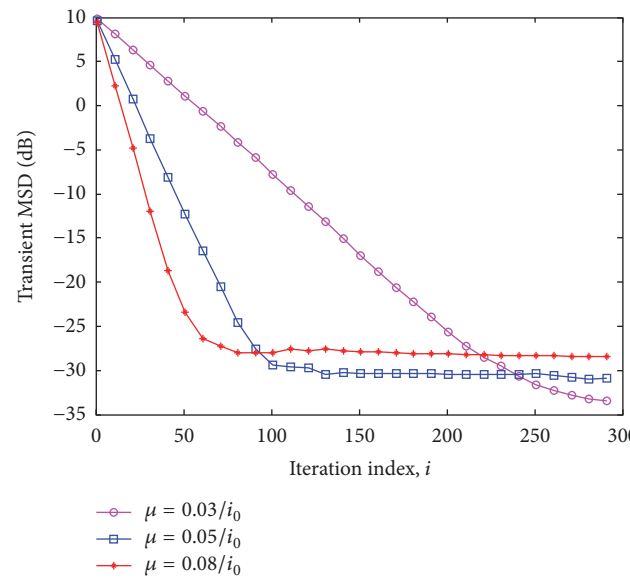

Figure 4: Transient network MSDs of the memory d-LMS for estimating the parameter ω_0 with different step sizes; $i_0 = 20$.

The weights of $a_{l,k}$ and $c_{l,k}$ are defined according to the relative degree and Metropolis rule [10, 13]: that is,

$$a_{l,k} = \begin{cases} \dfrac{\deg_l}{\sum_{n \in N_k} \deg_n}, & \text{if } l \in N_k \\ 0, & \text{otherwise,} \end{cases}$$

$$c_{l,k} = \begin{cases} \dfrac{1}{\max\left\{\deg_l, \deg_k\right\}} & \text{if } l \in N_k,\ l \neq k \\ 1 - \displaystyle\sum_{l \neq k} c_{l,k} & \text{if } l = k \\ 0, & \text{otherwise,} \end{cases} \tag{43}$$

where $\deg_l$ denotes the number of neighbors of node l or the degree of node l.

We show the transient network mean square deviation (MSD): $\sum_{k=1}^{K} E(\beta_k - \beta_{k,i})^2/K$ with different step-sizes μ and $i_0 = 20$ in Figure 4. Figure 4 illustrates that the estimate is convergent. As observed, the smaller μ is, the more accurate the estimate is (the slower the convergence rate).

Figure 5 demonstrates the network transient behavior in terms of mean square deviation (MSD): $\sum_{k=1}^{K} E(\beta_k - \beta_{k,i})^2/K$ for the proposed memory d-LMS algorithm, d-LMS algorithm [9], and memory global LMS (3). The values are obtained by averaging over 200 independent experiments. As the results indicate, the performance of the proposed memory d-LMS is closed to that of the global solution. We also observe that the performance of the proposed algorithm exceeds that of d-LMS [9] by more than 4 dB.

Figure 6 shows tracking results of the changes in the parameter for the memory d-LMS with different i_0 and traditional d-LMS. Additionally, step size of memory d-LMS is $0.08/i_0$. Step size of d-LMS is 0.08. In the beginning, $\omega_0 = [-1, 3]^T$, and after 200 iterations, it is set to $[1, 5]^T$. It can be observed that the proposed tracing performance of the

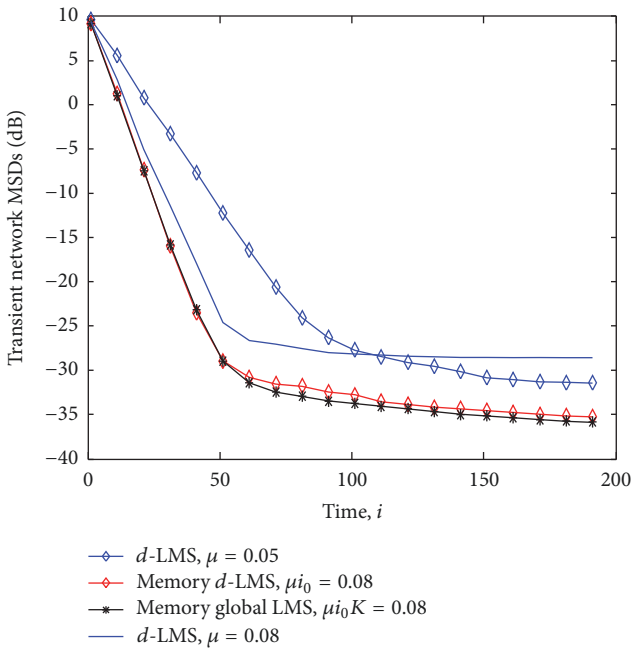

FIGURE 5: Transient network MSDs for different schemes with $i_0 = i$.

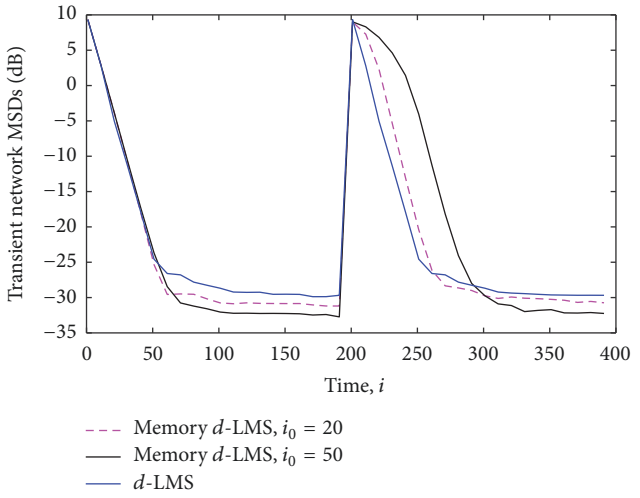

FIGURE 6: Transient network MSDs of the different algorithms for estimating the parameter. An abrupt change occurs at the 200th iteration.

memory d-LMS outperforms that of traditional d-LMS both with $i_0 = 20$ and $i_0 = 50$. It also shows that the larger i_0 is, the more accurate the estimate is. This is because that more statistical estimation information is considered in the estimation algorithm.

As observed, the smaller μ is, the more accurate the estimate is and the slower the convergence rate is. Hence, it is necessary to choose the small μ and the large i_0 according to (34) for systems that have strict request on estimate accuracy but are insensitive to convergence rate. Otherwise, choose the large μ and the small i_0 according to (34).

5. Conclusion

In this paper, a distributed estimation algorithm called memory d-LMS is derived and analyzed. The proposed algorithm consists of three steps. Firstly, at time i, for each node, a set of neighbor nodes collect and process their local information from time $i - i_0 + 1$ to time i, to obtain their preestimates. Secondly, all nodes transmit the preestimates to their neighbors. Finally, each node combines the collected information together with its local preestimate to generate improved estimates. Simulation results show that the proposed algorithm improves the convergence rate and reduces the MSE effectively.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported in part by the Natural Science Foundations of Zhejiang Province (LQ15F010004) and by the National Natural Science Foundations of China (61501158).

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdury, "Wireless multimedia sensor networks: a survey," *IEEE Wireless Communications Magazine*, vol. 14, no. 6, pp. 32–39, 2007.

[2] Z. A. Khan, E. Pignaton De Freitas, T. Larsson, and H. Abbas, "A multi-agent model for fire detection in coal mines using wireless sensor networks," in *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013*, pp. 1754–1761, Melbourne, Australia, July 2013.

[3] W. Zhang and T. Jiang, "Intrusion detection and classification in forest area using inter-sensor communication signals and SVM," in *Proceedings of the 2014 IEEE International Conference on Communication Problem-Solving, ICCP 2014*, pp. 401–404, Beijing, China, December 2014.

[4] J. Song, S. Han, A. K. Mok et al., "WirelessHART: applying wireless technology in real-time industrial process control," in *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '08)*, pp. 377–386, St. Louis, Mo, USA, April 2008.

[5] Y. He, D. Lu, and Y. Peng, "State estimation for multilevel multisensor data fusion systems," *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 27, no. 8, pp. 60–63, 1999.

[6] Q. Gan and C. J. Harris, "Comparison of two measurement fusion methods for Kalman-filter-based multisensor data fusion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 1, pp. 273–280, 2001.

[7] H. Chen, T. Kirubarajan, and Y. Bar-Shalom, "Comparison of centralized and distributed tracking algorithms using air to air scenarios," *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, vol. 4048, pp. 440–451, 2000.

[8] S. Xu, B. Lu, M. Baldea et al., "Data cleaning in the process industries," *Reviews in Chemical Engineering*, vol. 31, no. 5, pp. 453–490, 2015.

[9] C. Li, P. Shen, Y. Liu, and Z. Zhang, "Diffusion information theoretic learning for distributed estimation over network,"

IEEE Transactions on Signal Processing, vol. 61, no. 16, pp. 4011–4024, 2013.

[10] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, 2013.

[11] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, part 1, pp. 1035–1048, 2010.

[12] F. S. Cattivelli, C. G. Lopes, and A. . Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, 2008.

[13] Z. Liu, C. Li, and Y. Liu, "Distributed censored regression over networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5437–5449, 2015.