SELECTIVE PARTIAL UPDATE NORMALIZED LEAST MEAN SQUARE ALGORITHMS FOR DISTRIBUTED ESTIMATION OVER AN ADAPTIVE INCREMENTAL NETWORK

Mohammad Shams Esfand Abadi, and Ali-Reza Danaee

Faculty of Electrical and Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran
Emails: mshams@srttu.edu, ar.danaee@srttu.edu

Abstract: Selective partial update (SPU) strategy in adaptive filter algorithms is used to reduce the computational complexity. In this paper we apply the SPU Normalized Least Mean Squares algorithms (SPU-NLMS) for distributed estimation problem based on incremental strategy in a incremental network. The distributed SPU-NLMS (dSPU-NLMS) reduces the computational complexity while it's performance is close to the dNLMS. We demonstrate the good performance of dSPU-NLMS in both convergence speed and steady-state mean square error.

***Keywords*:** Selective partial update, normalized least mean squares, distributed estimation, incremental network.

I. INTRODUCTION

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. Obviously, the effectiveness of any distributed implementation will depend on the modes of cooperation that are allowed among the nodes [1].

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 [2].

Consensus implementations employ two time scales and they function 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 then combine their estimates through several consensus iterations; under suitable conditions, the estimates generally converge asymptotically to the desired (global) estimate of the parameter.

To use the incremental algorithms to propose an adaptive network structure composed of an interconnected set of nodes that is able to respond to data in real time and to track variations in the statistical properties of the data as follows: Each time a node receives a new piece of information, this information is readily used by the node to update its local estimate of the parameter of interest. Then the local estimates of the parameter are shared with the immediate neighbors of the node in a process that allows the information to flow to other nodes in the network [2]. The NLMS algorithms apply in estimation of adaptive filter with colored input. To reduce the computational complexity, in this paper we apply the SPU-NLMS algorithms [3], [4] in incremental adaptive network to establish of the dSPU-NLMS adaptive algorithm. We justified the performance of this algorithm in distributed network in both convergence speed and steady steady-state performance.

II. DSPU-NLMS ALGORITHM

Fig. 1 shows the learning algorithm in an incremental network for the dSPU-NLMS adaptive algorithms. Based on the local data $(d_k(n), x_k(n))$, and $h_{k-1}(n)$ from its previous node $(k-1)$ in the cycle, the update for local estimation is performed at each time instant n. Finally, the local estimation $h_Q(n)$ is used as the global estimation for filter coefficients. This value is used as the initial local estimation $h_1(n+1)$ for the next time instant $n+1$. The local filter vector update equation for the distributed normalized least mean square adaptive algorithms in an incremental network is given by

$$h_k(n) = h_{k-1}(n) + \frac{\mu_k}{\|\mathbf{x}_k(n)\|^2}\mathbf{x}_k(n)e_k(n) \tag{1}$$

where

$$e_k(n) = d_k(n) - \mathbf{x}_k^T(n)h_{k-1}(n) \tag{2}$$

is the error vector, and μ_k is the step-size of each node. Also, $\mathbf{x}_k(n) = [x_k(n), x_k(n-1), \cdots, x_k(n-M+1)]^T$ is the $M \times 1$ data signal vector and $d_k(n) = \mathbf{x}_k^T(n)h + v_k(n)$ where $v_k(n)$ is a temporally and spatially independent noise sequence with variance $\sigma_{v,k}^2$. In the following we introduce

the dSPU-NLMS adaptive algorithm. Now partition the input signal vector and the vector of filter coefficients into B blocks each of length L which are defined as

$$\mathbf{x}_k(n) = [x_{1,k}^T(n), x_{2,k}^T(n), \cdots, x_{B,k}^T(n)]^T \quad (3)$$

$$h_k(n) = [h_{1,k}^T(n), h_{2,k}^T(n), \cdots, h_{B,k}^T(n)]^T \quad (4)$$

Let us suppose that we wish to update S blocks out of B at every iteration. Let $\tau_S = \{i_1, i_2, \cdots, i_S\}$ denote a $S-$subset (subset with S members) of the set $\{1, 2, \cdots, , B\}$ and let Ψ be the collection of all $S-$subsets, i.e. $\tau_S \in \Psi$. The selection of blocks can be carried out by considering the following constrained minimization problem:

$$\min_{\tau_S \in \Psi} \min_{h_{\tau_S,k}(n)} \|h_{\tau_S,k}(n) - h_{\tau_S,k-1}(n)\|^2 \quad (5)$$

subject to

$$h_k^T(n)\mathbf{x}(n) = d_k(n) \quad (6)$$

where the $SL \times 1$ vector $h_{\tau_S,k}(n)$ is defined by

$$h_{\tau_S,k}(n) = [h_{i_1,k}^T(n), h_{i_2,k}^T(n), \cdots, h_{i_S,k}^T(n)]^T \quad (7)$$

If τ_S were given and fixed, (5) could be solved by minimizing the cost function

$$J_{\tau_{S,k}}(n) = \frac{1}{2}\|h_{\tau_S,k}(n) - h_{\tau_S,k-1}(n)\|^2 + \lambda(d_k(n) - h_k^T(n)\mathbf{x}_k(n)) \quad (8)$$

where λ is a Lagrange multiplier. Minimization of $J_{\tau_S}(n)$ with respect to and results in

$$h_{\tau_S,k}(n) = h_{\tau_S,k-1}(n) + \frac{1}{\|\mathbf{x}_{\tau_{S,k}}(n)\|^2}\mathbf{x}_{\tau_S,k}(n)e(n) \quad (9)$$

where $\mathbf{x}_{\tau_{S,k}}(n)$ is defined by

$$\mathbf{x}_{\tau_S,k}(n) = [x_{i_1,k}^T(n), x_{i_2,k}^T(n), \cdots, x_{i_S,k}^T(n)]^T \quad (10)$$

After the introduction of a small positive step-size μ, we obtain the dNLMS algorithm for the update of S blocks specified by τ_S.

The number of unique $S-$subsets of $\{1, 2, \cdots, B\}$ is

$$\binom{M}{S} = \frac{B!}{S!(B-S)!} \quad (11)$$

To determine which blocks to update, we need to find S coefficient blocks with the minimum squared-Euclidean-norm update. The block selection problem can be formally written as

$$\begin{aligned}\tau_S = & \quad arg \quad \min_{\tau_S \in \Psi} \|h_{J_S,k}(n) - h_{J_S,k-1}(n)\|^2 \\ & = arg \quad \min_{J_S \in \Psi} \|\frac{\mathbf{x}_{J_S,k}(n)e_k(n)}{\|\mathbf{x}_{J_S,k}(n)\|^2}\|^2 \\ & = arg \quad \max_{J_S \in \Psi} \|\mathbf{x}_{J_S,k}(n)\|^2 \\ & = arg \quad \max_{J_S \in \Psi} \sum_{J \in J_S} \|\mathbf{x}_{J,k}(n)\|^2 \end{aligned} \quad (12)$$

The optimum τ_S satisfying (12) is found simply by ranking the regressor vector blocks according to their squared Euclidean norms and choosing the S largest blocks, i.e

$$\begin{aligned}\|\mathbf{x}_{i_1,k}(n)\|^2 \geqslant \|\mathbf{x}_{i_2,k}(n)\|^2 \geqslant \cdots \geqslant \|\mathbf{x}_{i_S,k}(n)\|^2 \\ \geqslant \|\mathbf{x}_{j,k}\|^2 \qquad \forall_i \in \{1, 2, \cdots, B\}\end{aligned} \quad (13)$$

The recursive algorithm combining (8) and (12) is given by

$$\begin{aligned}h_{\tau_S,k}(n) = h_{\tau_S,k-1}(n) + \tfrac{\mu}{\|\mathbf{x}_{\tau_S,k}(n)\|^2}\mathbf{x}_{\tau_S,k}(n)e_k(n) \\ \tau_S = \{i : \|\mathbf{x}_{i,k}(n)\|^2 \; is \; one \; of \; the \; S \; largest \; among \\ \|\mathbf{x}_1(k)\|^2, \cdots, \|\mathbf{x}_B(k)\|^2\}\end{aligned} \quad (14)$$

which we will refer to as the distributed selective-partial-update NLMS (dSPU-NLMS) algorithm.

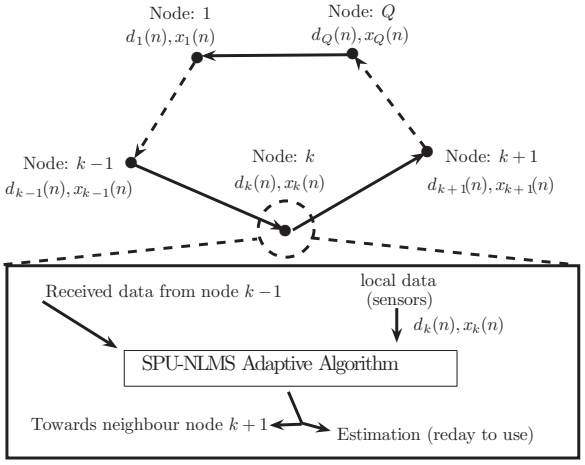

Fig. 1. Data Processing in an incremental network through SPU-NLMS adaptive algorithm

III. COMPUTATIONAL COMPLEXITY

Table 1, shows the number of multiplications, divisions, and comparisons of dNLMS, and dSPU-NLMS per iteration per node. For every input sample, the computational complexity of (12) is $M + SL + 2$ multiplications, one division, and $B - 1$ comparisons. If $B = M$ (i.e., $L = 1$), the maximum number of comparisons can be reduced to $[2\log_2^M] + 2$ by employing the sortline algorithm [4]. As we can see, the number of multiplications in dSPU-NLMS is lower than dNLMS algorithm.

IV. SIMULATION RESULTS

We justified the performance of dSPU-NLMS algorithm in an incremental network with $Q = 20$ nodes in a system identification setup. The impulse response of the random unknown system has $M = 16$ taps. The correlated input signal, $x_k(n)$, at each node are generated by passing a white Gaussian noise process with unit variance through the a first

order autoregressive model filter with the z-transfer function $\frac{\sqrt{1-\alpha_k^2}}{1-\alpha_k z^{-1}}$ and $\alpha_k \in [0, .5)$. The noise sequence of each node is a white Gaussian process with variance $\sigma_{v,k}^2 \in (0, 0.1]$. Fig. 2 shows the node profiles of $\sigma_{v,k}^2$ and α_k. In all simulations, the simulated MSE, and MSD learning curves are obtained by ensemble averaging over 200 independent trials. Also, the steady-state MSE, MSD and EMSE values are obtained by averaging over 1000 steady-state samples from 100 independent realizations for each value of μ for a given algorithm. The step-sizes were chosen to get approximately the same steady-state MSE, and MSD in all learning curves. In dSPU-NLMS algorithm, the parameter B was set to 4, and different values for S have been chosen in simulations. Fig. 3 presents the simulated learning curves of dNLMS, and dSPU-NLMS with different values of S for node 10. As we can see for large values of S, the performance of dSPU-NLMS will be close to the dNLMS. Fig. 4 compares the performance of dSPU-NLMS, and dNLMS for node 10 based on MSD curves. Again close performance can be seen for dSPU-NLMS especially for $S = 3$. Fig. 5-7 show the simulated steady-state MSE, MSD and EMSE values of dSPU-NLMS for each node with $\mu_k = 0.5$. The simulation results show that the steady-state values of dSPU-NLMS algorithm are close to steady-state values of dNLMS for $S = 2$, and 3.

Fig. 8-10 show the steady-state MSE, MSD and EMSE of dSPU-NlMS as function of the step-size for node 10. The step-sizes change in the range [0.008, 0.5] which are in the stability bound of dSPU-NlMS. Again the steady state values of the dSPU-NLMS for different step-sizes are close to the dNLMS values. For $S = 2$, and 3, the better agreement can be seen.

V. CONCLUSION

In this paper we applied the SPU-NLMS algorithm in distributed estimation in an incremental network. The dSPU-NLMS has close performance to dNLMS algorithm. The computational complexity of this algorithm is lower than dNLMS because of partial update approach. Simulation results showed the good performance of this algorithm in an incremental network for steady state error and convergence speed.

Table I. Computational complexity of dNLMS, and dSPU-NLMS adaptive algorithm per iteration per node

Algorithm	NLMS	SPU-NLMS
Multiplications	$2M+2$	$M+SL+2$
Divisions	1	1
Comparisons	–	$O(B)+Blog_2S$

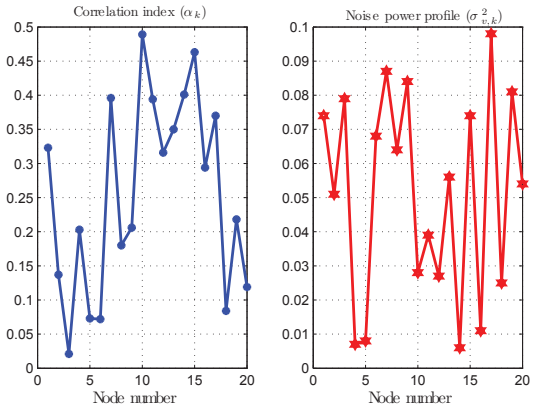

Fig. 2. Node profile: Noise power and Correlation index for the Gaussian data Network.

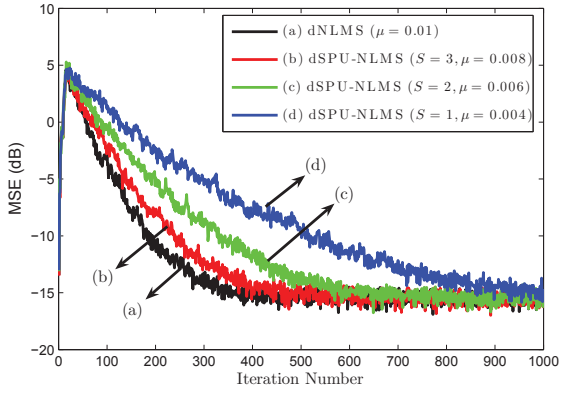

Fig. 3. The MSE learning curves of dNLMS and dSPU-NLMS adaptive algorithm

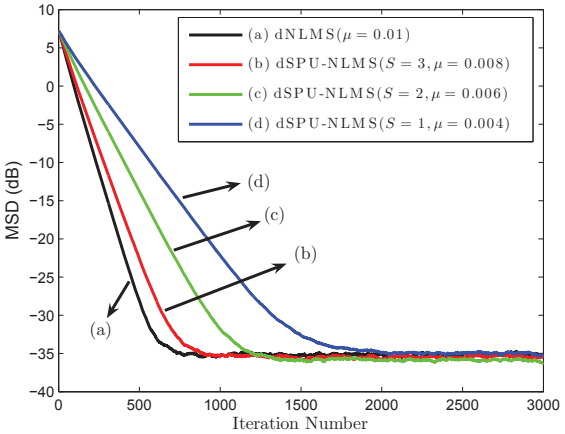

Fig. 4. The MSD learning curves of dNLMS and dSPU-NLMS adaptive algorithm

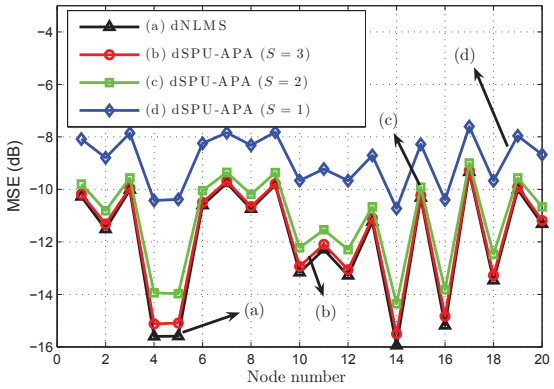

Fig. 5. Steady-state MSE for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$

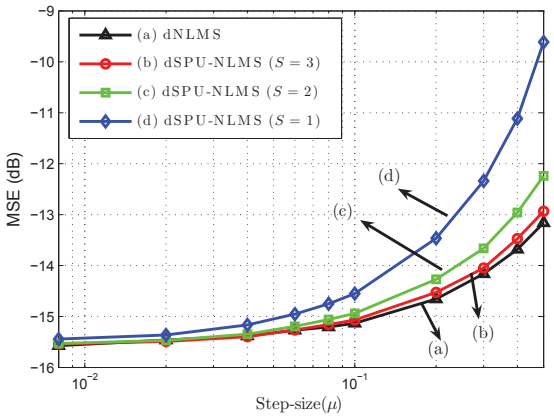

Fig. 8. Steady-state MSE curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size.

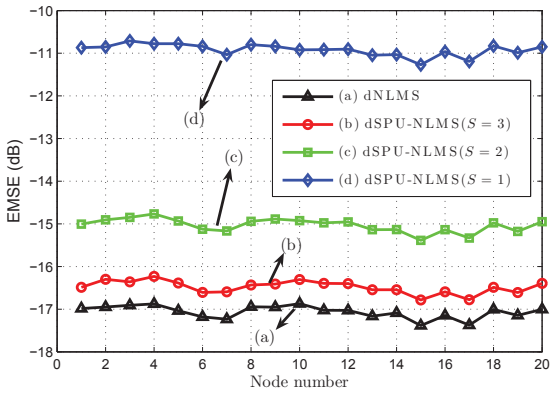

Fig. 6. Steady-state EMSE for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$.

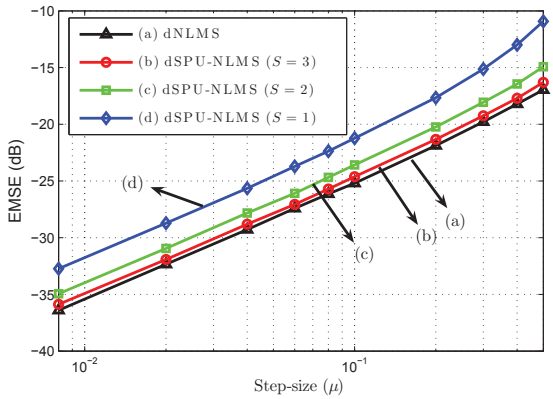

Fig. 9. Steady-state EMSE curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size.

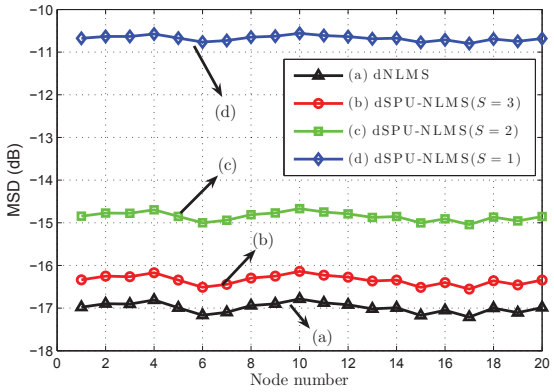

Fig. 7. Steady-state MSD for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$.

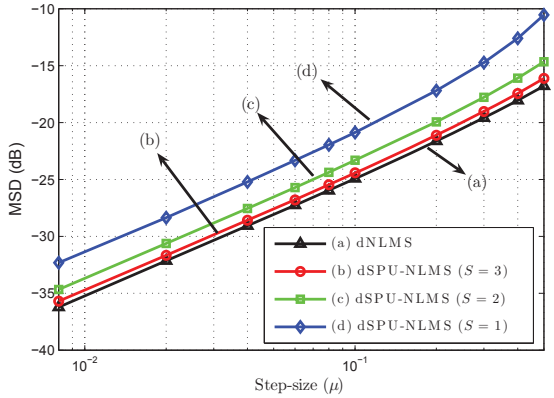

Fig. 10. Steady-state MSD curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size.

VI. REFERENCES

[1] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Processing*, vol. 55, pp. 4064–4077, 2007.

[2] L. Li, J. A. Chambers, C. G. Lopes, and A. H. Sayed, "Distributed estimation over an adaptive incremental network based on the affine projection algorithm," *IEEE Trans. Signal Processing*, vol. 58, pp. 151–164, 2010.

[3] K. Doğançay, *Partial-Update Adaptive Signal Processing, Design Analysis and implementation*. Academic Press, 2009.

[4] K. Dogancay and O. Tanrikulu, "Adaptive filtering algorithms with selective partial updates," *IEEE Trans. Circuits, Syst. II: Analog and Digital Signal Processing*, vol. 48, pp. 762–769, 2001.