ADJUSTMENT OF COMBINATION WEIGHTS OVER ADAPTIVE DIFFUSION NETWORKS

Jesus Fernandez-Bes, Jerónimo Arenas-García

Signal Theory and Communicatios Dept.
Universidad Carlos III de Madrid
Av. Universidad 30, 28911, Leganes (Spain)

Ali H. Sayed

Electrical Engineering Dept.
University of California, Los Angeles
Los Angeles, CA 90095

ABSTRACT

We show how the convergence time of an adaptive network can be estimated in a distributed manner by the agents. Using this procedure, we propose a distributed mechanism for the nodes to switch from using fixed doubly-stochastic combination weights to adaptive combination weights. By doing so, and by knowing when to switch, the agents are able to enhance their steady-state mean-square-error performance without degrading the rate of convergence during the transient phase of the learning algorithm.

Index Terms— Adaptive networks, convergence time, distributed estimation, diffusion networks.

1. INTRODUCTION AND RELATED WORK

In adaptive networks a collection of agents cooperate with each other through local interactions in order to solve a distributed inference problem in real-time, such as distributed estimation or tracking of parameters of interest [1, 2]. Each node in the adaptive network relies on the fusion of information collected from its local neighbors. Different static combination rules have been proposed in the literature such as uniform [3], Metropolis [4], or relative degree [2, 5] rules. In addition, several schemes for adapting the combination weights have been proposed [2, 6, 7] to enhance the mean-square-error performance of the learning process. However, experiments and analysis in [8] show that the better steady-state performance comes at the expense of a slower convergence rate during the transient phase. This is because the process of adapting the combination weights degrades the speed of learning during the initial stages of adaptation. There have been useful studies in the literature on accelerating the convergence rate of distributed strategies (e.g., [4, 9, 10]). However, these works focus on traditional consensus implementations for computing averages and do not deal directly with the problem of real-time adaptation from streaming data. In this case, it is not sufficient to focus solely on the convergence rate during the transient phase of the algorithm, it is also critical to combine this analysis with the performance of the algorithm in steady state. This is important because adaptive solutions need to satisfy two conditions: to learn well and to track well. We examine this question in the context of diffusion networks since these have been shown to have superior stability and performance ranges in comparison to consensus networks for the case of adaptation over networks [11].

In this regard, a technique was proposed in [8] to hold the combination weights fixed for an initial period of time and then switch to adaptive combination weights. A hypothesis testing problem was formulated in [8] that enables the individual agents to decide whether the network has approached steady state or not. Depending on the local conditions at each node, which are affected by local SNR values, some of the agents may decide in favor of switching earlier than needed, which can cause some deterioration in convergence speed.

In this work, we propose an alternative switching criterion based on allowing the agents to assess, in a distributed fashion, the convergence time of the network. This is the time at which the network performance gets close to its steady-state level. We first derive an expression for the network convergence time, and then explain how it can be evaluated through local cooperation by the agents. This time can be estimated in advance before the network reaches steady-state. We plot a histogram illustrating the time instants at which the agents switch in comparison to the actual convergence time. Results show that the histogram is approximately centered around the actual value with a small spread, which is a desirable feature. We compare the performance of the proposed scheme to earlier schemes and illustrate its superior performance.

2. DIFFUSION STRATEGY

Consider a connected network consisting of N nodes. The set of neighbors of node k, including the node itself, is denoted by $\mathcal{N}_k$. At every time instant i, every node k has access to a scalar measurement $d_k(i)$ and a regression row vector $u_{k,i}$ of size M, both realizations of zero-mean random processes $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$. We use boldface symbols to refer to random variables and normal font to refer to their realizations. We assume that the measurements are related to some unknown column vector w^o of size M and unit norm through a linear regression model of the form:

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^o + \boldsymbol{v}_k(i), \quad \|w^o\| = 1 \tag{1}$$

where $\boldsymbol{v}_k(i)$ denotes measurement noise and is assumed to be a zero-mean white random process with power $\sigma^2_{v,k}$. The regression data $\{\boldsymbol{u}_{k,i}\}$ are assumed to be temporally white and independent over space with uniform covariance matrix $R_u = \mathbb{E}\boldsymbol{u}^*_{k,i}\boldsymbol{u}_{k,i} > 0$ for all $k = 1, 2, \ldots, N$. Moreover, the noise process $\boldsymbol{v}_\ell(j)$ is assumed to be independent over space and also independent of $\boldsymbol{u}_{k,i}$ for all k, ℓ, i, j.

The objective of the network is to estimate w^o in a distributed manner through collaboration among the nodes by minimizing the aggregate cost function:

The work of Fernandez-Bes and Arenas-García was partly supported by MINECO projects TEC2011-22480 and PRI-PIBIN-2011-1266. The work of Fernandez-Bes was also supported by Spanish MECD FPU program. The work of A. H. Sayed was supported in part by NSF grant CCF-1011918. This work was performed while J. Fernandez-Bes was a visiting graduate student at the UCLA Adaptive Systems Laboratory.
Emails:{jesusfbes,jarenas}@tsc.uc3m.es,sayed@ee.ucla.edu

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

$$J^{\rm glob}(w) = \sum_{k=1}^{N} \mathbb{E}|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} w|^2 \quad (2)$$

Several adaptive diffusion strategies have been developed for this purpose [1, 2, 12]. Without loss of generality, in this work, we consider the Adapt-then-Combine (ATC) form where the estimate $\boldsymbol{w}_{k,i}$ for w^o at node k and time i is updated as follows:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu \boldsymbol{u}_{k,i}^* [\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{w}_{k,i-1}] \quad (3)$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell,k} \boldsymbol{\psi}_{\ell,i} \quad (4)$$

where the $\{a_{\ell,k}\}$ are non-negative convex combination coefficients. We collect these coefficients into an $N \times N$ matrix $A = [a_{\ell,k}]$. Then, A is a left-stochastic matrix such that

$$A^T \mathbb{1} = \mathbb{1}, \quad a_{\ell,k} = 0 \text{ if } \ell \notin \mathcal{N}_k \quad (5)$$

where $\mathbb{1}$ denotes the vector with all its entries equal to one. In the first step (3), an intermediate estimate $\boldsymbol{\psi}_{k,i}$ is computed following adaptation based on local data at node k. All other nodes in the network perform a similar update simultaneously. The second step (4) aggregates the intermediate estimates from the neighborhood of node k and generates $\boldsymbol{w}_{k,i}$. The process is repeated continuously by all nodes using a positive constant step-size parameter, μ.

We introduce the weight-error vectors, $\tilde{\boldsymbol{w}}_{k,i} = w^o - \boldsymbol{w}_{k,i}$, and collect all errors from across the network into the column vector, $\tilde{\boldsymbol{w}}_i = \text{col}\{\tilde{\boldsymbol{w}}_{1,i}, \cdots, \tilde{\boldsymbol{w}}_{N,i}\}$. It can be shown that, under expectation and for sufficiently small step-sizes, the weight-error dynamics of the network evolves according to the following mean-square-error recursion [1, 2]:

$$\mathbb{E}\|\tilde{\boldsymbol{w}}_i\|^2 = \mathbb{E}\|\tilde{\boldsymbol{w}}_{i-1}\|^2_{\mathcal{B}^* \mathcal{B}} + \text{Tr}(\mathcal{Y}) \quad (6)$$

where

$$\mathcal{B} \triangleq A^T \otimes (I_M - \mu R_u) \quad (7)$$

$$\mathcal{A} \triangleq A \otimes I_M \quad (8)$$

$$\mathcal{S} \triangleq \text{diag}\{\sigma_{v,1}^2, \ldots, \sigma_{v,N}^2\} \otimes R_u \quad (9)$$

$$\mathcal{Y} \triangleq \mu^2 \mathcal{A}^T \mathcal{S} \mathcal{A} \quad (10)$$

where $\otimes$ denotes the matrix Kronecker product. Relation (6) is very useful because it allows us to analyze both the steady-state and transient behavior of the network. The steady-state performance of the network can be measured in terms of its mean-square-deviation (MSD) defined as:

$$\text{MSD} \triangleq \lim_{i \to \infty} \frac{1}{N} \sum_{k=1}^{N} \mathbb{E}\|\tilde{\boldsymbol{w}}_{k,i}\|^2 \quad (11)$$

When the combination weights are fixed throughout the adaptation process, various expressions and approximations have been derived in the literature for the above MSD. For example, when A is a doubly-stochastic matrix, meaning that each of its columns adds up to one and each of its rows also adds up to one, then the MSD is approximately given by [2]:

$$\text{MSD} \approx \frac{\mu M}{2N} \bar{\sigma}_v^2 = \frac{\mu M}{2N} \left(\frac{\sum_{k=1}^{N} \sigma_{v,k}^2}{N} \right) \quad (12)$$

in terms of the average noise power across the network.

When the combination policy is left-stochastic, it is possible to design adaptive strategies that optimize the MSD over the combination weights [1, 6, 7]. However, it was noted in [8] that these strategies tend to slow down the convergence rate during the transient phase due to the additional burden of learning the combination coefficients. In the next section we propose a method to calculate the convergence time in a distributed manner and use this method to allow the nodes to switch between two fixed combination strategies: one for the initial transient phase and another for the later stages of adaptation.

3. CONVERGENCE TIME

We observe from (6) that during the transient phase of the algorithm, the convergence rate of the network is determined by $[\rho(\mathcal{B})]^2$, which is the square of the spectral radius of $\mathcal{B}$. This spectral radius is smaller than one when the algorithm is mean-square-stable, which is satisfied for sufficiently small values of the step-size parameter [1, 2]. From (6), and from the Rayleigh-Ritz characterization of eigenvalues [13, 14] we note that

$$\mathbb{E}\|\tilde{\boldsymbol{w}}_i\|^2 \leq [\rho(\mathcal{B})]^2 \mathbb{E}\|\tilde{\boldsymbol{w}}_{i-1}\|^2 + \text{Tr}(\mathcal{Y}) \quad (13)$$

We define the convergence time of the network as the number of iterations, T, that is needed for its mean-square-error to reach $1 + \epsilon$ times its steady-state MSD value, for some small given $\epsilon > 0$. That is, the time T satisfies:

$$\frac{1}{N} \mathbb{E}\|\tilde{\boldsymbol{w}}_T\|^2 = (1 + \epsilon) \text{MSD} \quad (14)$$

Using a procedure similar to [15, Chap. 23.A], we can derive an expression for the convergence time of the network as follows. We start by getting an upper bound on the MSD from (13), using the fact that once converged, $\mathbb{E}\|\tilde{\boldsymbol{w}}_i\|^2 = \mathbb{E}\|\tilde{\boldsymbol{w}}_{i-1}\|^2$:

$$\text{MSD}_{\rm UB} = \frac{\text{Tr}(\mathcal{Y})}{N(1 - [\rho(\mathcal{B})]^2)} \quad (15)$$

Then, we center the mean-square-error around its steady-state bound:

$$\begin{aligned} \mathbb{E}\|\tilde{\boldsymbol{w}}_i\|^2 - N \cdot \text{MSD}_{\rm UB} &\leq \rho(\mathcal{B})]^2 \mathbb{E}\|\tilde{\boldsymbol{w}}_{i-1}\|^2 + \text{Tr}(\mathcal{Y}) \\ &\quad - N \cdot \text{MSD}_{\rm UB} \\ &\leq [\rho(\mathcal{B})]^2 \left(\mathbb{E}\|\tilde{\boldsymbol{w}}_{i-1}\|^2 - N \cdot \text{MSD}_{\rm UB} \right) \end{aligned} \quad (16)$$

where the second inequality was obtained by replacing $\text{Tr}(\mathcal{Y})$ using expression (15). Iterating over this recursion we get:

$$\mathbb{E}\|\tilde{\boldsymbol{w}}_i\|^2 - N \cdot \text{MSD}_{\rm UB} \leq [\rho(\mathcal{B})]^{2i} \left(\mathbb{E}\|\tilde{\boldsymbol{w}}_0\|^2 - N \cdot \text{MSD}_{\rm UB} \right) \quad (17)$$

where $\mathbb{E}\|\tilde{\boldsymbol{w}}_0\|^2 = \mathbb{E}\|w^o - \boldsymbol{w}_{-1}\|^2$. If we assume $\boldsymbol{w}_{-1} = 0$ then $\mathbb{E}\|\tilde{\boldsymbol{w}}_0\|^2 = \|w^o\|^2 = 1$. Finally, using the definition of convergence time and solving for T we get

$$T \leq \frac{\ln \left(\frac{\epsilon N \text{MSD} + N \cdot (\text{MSD} - \text{MSD}_{\rm UB})}{1 - N \cdot \text{MSD}_{\rm UB}} \right)}{2 \ln (\rho(\mathcal{B}))} \quad (18)$$

From (7), and assuming sufficiently small step-sizes, we have $\rho(\mathcal{B}) = 1 - \mu \lambda_{\min}(R_u)$. In addition, since we are interested in approximating the convergence time, we may assume that the upper bound given by (15) is tight and set $\text{MSD} \approx \text{MSD}_{\rm UB}$ in (12) so that

$$T \approx \frac{\ln\left(\frac{\epsilon N \cdot \text{MSD}}{1-N\cdot\text{MSD}}\right)}{2\ln\left(1-\mu\lambda_{\min}(R_u)\right)} \tag{19}$$

where we further use expression (12) for the MSD. Finally, since we can avoid the need for computing $\lambda_{\min}(R_u)$ by approximating it by $\text{Tr}(R_u)/M$, which amounts to using the average value of the eigenvalues of R_u,

$$T \approx \frac{\ln\left(\frac{\epsilon N \cdot \text{MSD}}{1-N\cdot\text{MSD}}\right)}{2\ln\left(1-\mu\text{Tr}(R_u)/M\right)} \tag{20}$$

In the next section we explain how this convergence time can be evaluated in a distributed and adaptive manner by each agent.

4. DISTRIBUTED CONVERGENCE TIME ESTIMATION

Observe that in (20) there are two terms the nodes need to estimate to be able to compute the convergence time: MSD from (12) and $\text{Tr}(R_u)$.

To estimate the MSD, the nodes need to estimate the average noise variance $\bar{\sigma}_v^2$ in (12). This value can be estimated using a diffusion construction as follows:

$$\boldsymbol{\theta}_k(i) = (1-\alpha_v)\widehat{\boldsymbol{\sigma}}_k^2(i-1) + \alpha_v|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1}|^2 \tag{21}$$

$$\widehat{\boldsymbol{\sigma}}_k^2(i) = \sum_{\ell\in\mathcal{N}_k} c_{\ell,k}\boldsymbol{\theta}_k(i) \tag{22}$$

where $\alpha_v \in (0,1)$ is a forgetting factor. In the first step we adapt the estimate of the noise variance at node k, $\widehat{\boldsymbol{\sigma}}_k^2(i-1)$, using a local quadratic error to get the intermediate estimate, $\boldsymbol{\theta}_k(i)$. In the second step we combine the intermediate value using combination weights $c_{\ell,k}$ that can be chosen according to a doubly-stochastic rule such as the Metropolis rule [1] defined further ahead in (27).

On the other hand, $\text{Tr}(R_u)$ can be estimated locally by each node as follows:

$$\boldsymbol{\beta}_k(i) = (1-\alpha_\lambda)\boldsymbol{\beta}_k(i-1) + \alpha_\lambda\|\boldsymbol{u}_{k,i}\|^2 \tag{23}$$

with learning rate $\alpha_\lambda \in (0,1)$. Then, the estimate of the convergence time (20) by agent k at time i is given by

$$\boldsymbol{T}_k(i) = \frac{\ln\left(\frac{\epsilon\delta_k(i)}{1-\delta_k(i)}\right)}{2\ln\left(1-\frac{\mu\boldsymbol{\beta}_k(i)}{M}\right)} \tag{24}$$

where $\delta_k(i) = \frac{\mu M}{2}\widehat{\boldsymbol{\sigma}}_k^2(i)$ is the estimate of $N\cdot\text{MSD}$ by node k at time i.

5. PHASE SWITCHING ALGORITHM

It was argued in [6, 7] that enhanced MSD performance is obtained if nodes employ optimized combination coefficients $\{a_{\ell,k}\}$ chosen according to the relative-variance (ARV) rule:

$$\boldsymbol{a}_{\ell,k}(i) = \begin{cases} \frac{\boldsymbol{\gamma}_{\ell,k}^{-2}(i)}{\sum_{j\in\mathcal{N}_k}\boldsymbol{\gamma}_{j,k}^{-2}(i)}, & \text{if } \ell\in\mathcal{N}_k \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

where the scalars $\{\boldsymbol{\gamma}_{\ell,k}(i)\}$ are updated as follows using a learning factor $\alpha_a \in (0,1)$:

$$\boldsymbol{\gamma}_{\ell,k}^2(i) = (1-\alpha_a)\boldsymbol{\gamma}_{\ell,k}^2(i-1) + \alpha_a\|\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}\|^2 \tag{26}$$

However, it was noted in [8] that the improvement in MSD comes at the expense of deterioration in the convergence speed during the transient phase of the algorithm. It was therefore suggested there that it would be preferred for the agents to employ a fixed combination rule during the initial stages of adaptation before switching to using the adaptive combination (25). For this procedure to be feasible, it is necessary to endow the agents with the ability to detect when switching should occur. Now that we know how the agents can assess the convergence time of the algorithm in a distributed manner, it becomes possible to carry out this switching in an efficient manner: each agent can estimate T using (24) and switch to the adaptive combination rule at that time. During the transient stage, nodes can employ any doubly-stochastic rule (25), such as the Metropolis rule defined by:

$$a_{\ell,k} = \begin{cases} 1/\max\{n_k, n_\ell\}, & \text{if } k\neq\ell \text{ are neighbors} \\ 1 - \sum_{m\in\mathcal{N}_k\backslash\{k\}} a_{m,k}, & \text{if } k=\ell \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

where n_k denotes the degree of agent k, i.e., the number of its neighbors, and $\mathcal{N}_k\backslash\{k\}$ denotes the set of neighbors of node k excluding k itself. The proposed diffusion algorithm with switching is summarized in the listing bellow.

Algorithm 1: Diffusion adaptation with weight switching

Initialization : $\widehat{\boldsymbol{\sigma}}_k^2(-1), \boldsymbol{\gamma}_{\ell,k}(-1), \boldsymbol{\beta}_k(-1) = 0$;
$\boldsymbol{w}_{k,-1}, \boldsymbol{\psi}_{k,-1} = 0_M$;$\boldsymbol{T}_k(0) =$ positive integer $\gg 1$.
Start with Metropolis rule (27) for $\{a_{\ell,k}\}$ and $\{c_{\ell,k}\}$.
for each time $i \geq 0$ and each node k **do**
  sense $\boldsymbol{u}_{k,i}$ and $\boldsymbol{d}_k(i)$
  $\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu\boldsymbol{u}_{k,i}^*\left[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1}\right]$
  if $i < \boldsymbol{T}_k(i)$ **then**
    $\boldsymbol{\beta}_k(i) = (1-\alpha_\lambda)\boldsymbol{\beta}_k(i-1) + \alpha_\lambda\|\boldsymbol{u}_{k,i}\|^2$
    $\boldsymbol{\theta}_k(i) = (1-\alpha_v)\widehat{\boldsymbol{\sigma}}_k^2(i) + \alpha_v|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1}|^2$
    $\widehat{\boldsymbol{\sigma}}_k^2(i) = \sum_{\ell\in\mathcal{N}_k} c_{\ell,k}\boldsymbol{\theta}_k(i)$
    $\delta_k(i) = \frac{\mu M}{2}\widehat{\boldsymbol{\sigma}}_k^2(i)$ and $\boldsymbol{T}_k(i) = \frac{\ln\left(\frac{\epsilon\delta_k(i)}{1-\delta_k(i)}\right)}{2\ln\left(1-\frac{\mu\boldsymbol{\beta}_k(i)}{M}\right)}$
  else
    $\boldsymbol{\gamma}_{\ell,k}^2(i) = (1-\alpha_a)\boldsymbol{\gamma}_{\ell,k}^2(i-1) + \alpha_a\|\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}\|^2$
    $\boldsymbol{a}_{\ell,k} = \frac{\boldsymbol{\gamma}_{\ell,k}^{-2}(i)}{\sum_{j\in\mathcal{N}_k}\boldsymbol{\gamma}_{j,k}^{-2}(i)}$, if $\ell\in\mathcal{N}_k$
  end
  $\boldsymbol{w}_{k,i} = \sum_{\ell\in\mathcal{N}_k}\boldsymbol{a}_{\ell,k}\boldsymbol{\psi}_{k,i}$
end

Note that the convergence time computation is only carried out during the transient stage. During this phase, nodes need to communicate with the neighbors their scalar estimates $\boldsymbol{\theta}_k(i)$ in addition to $\boldsymbol{\psi}_{k,i}$, which amounts to only a slight increment in communication complexity.

6. SIMULATION RESULTS

We compare our proposed algorithm with the static Metropolis rule [4], the adaptive relative-variance (ARV) rule [6], and the phase detection algorithm of [8]. The topology of the 20-node simulated network is shown in Fig. 1. Vector w^o has length $M = 8$ and its entries are uniformly chosen in the range $[-1, 1]$ and normalized to $\|w^o\| = 1$. Regressors $\boldsymbol{u}_{k,i}$ are zero-mean Gaussian vectors with a common diagonal covariance R_u with entries generated between $[1, 2]$. The additive noise $\boldsymbol{v}_k(i)$ is also temporally white and spatially independent Gaussian with zero mean and different variances $\sigma^2_{v,k}$, generated uniformly in the range $[-30, 0]$ (dB). We assume that all nodes use step-size $\mu = 0.001$.

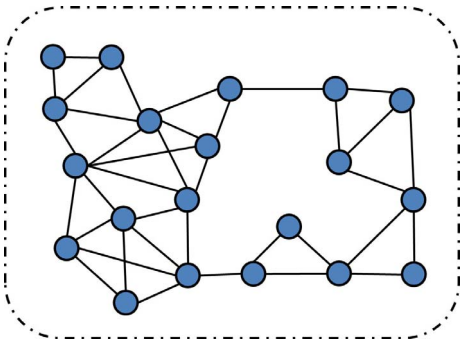
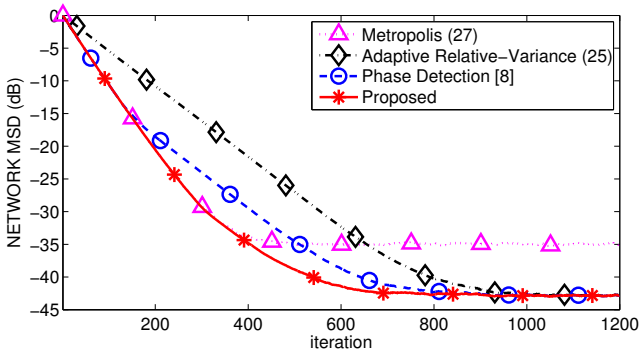


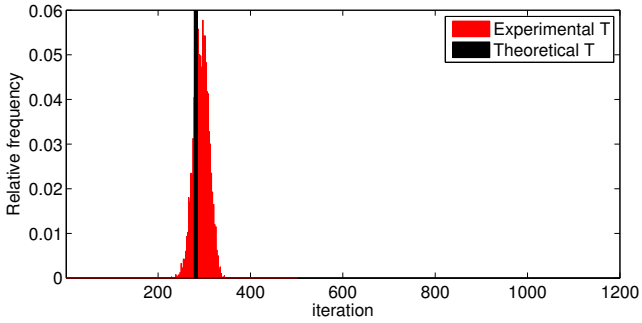
Fig. 1. Network topology for simulation.

Parameter α_a in the adaptive relative-variance update (26) is set to 0.5. For the algorithm in [8] we set $L = 60$. Regarding our algorithm, we set $\alpha_v = 0.2$, $\alpha_\lambda = 0.01$ and compute the convergence time corresponding to $\epsilon = 0.05$. The network MSD performance, averaged over 200 independent simulations, is shown in Fig. 2.(a). In addition, in Fig. 2.(b) we present the theoretical convergence time, calculated using (20), and a histogram of the switching times, $\boldsymbol{T}_k(i)$, of the nodes for all simulations. Finally, in Fig. 2.(c), the average learning curve of $\boldsymbol{T}_k(i)$ is shown.

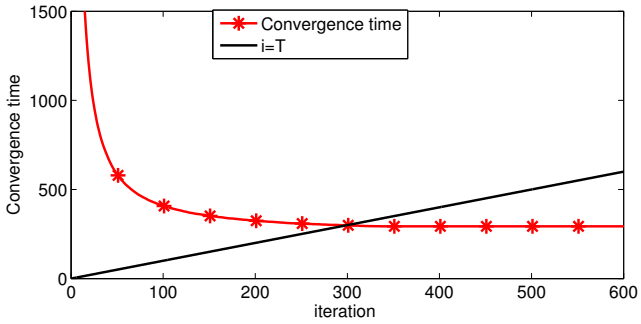
It can be observed that the estimation of convergence time $\boldsymbol{T}_k(i)$ is quite accurate: small bias and variance, and it converges fast enough. As can be seen in Fig. 2.(c), well before the switching time of switching the estimation of T has converged to an stable value. Consequently the phase switch is performed around the time it needs to be performed.



(a) Simulated network MSD curves.



(b) Histogram of switching times and optimal convergence time.



(c) The red curve represents the learning curve of the convergence time and its values are computed by averaging the estimates $\boldsymbol{T}_k(i)$ over all nodes for each time i. Black line represents $i = T$ shown to check if convergence time estimation converges fast enough.

Fig. 2. Performance of proposed strategy compared to adaptive relative-variance (ARV) rule [6] and Metropolis rule [4].

7. CONCLUSIONS

We proposed a method to estimate the convergence time of an adaptive diffusion network in a distributed manner. We then used this procedure to propose an algorithm to switch the operation of the network from a fixed doubly-stochastic rule to an adaptive combination rule. The switching enhances MSD performance in steady-state and improves convergence speed during the transient phase.

8. REFERENCES

[1] A. H. Sayed, "Diffusion adaptation over networks," *Academic Press Library in Signal Processing*, vol. 3, R. Chellapa and S. Theodoridis, *Eds.*, pp. 323–454, 2014, Also available as arXiv:1205.4220 [cs.MA], May 2012.

[2] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 155–171, May 2013.

[3] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in Proc. *IEEE Conf. Decision and Control*, Sevilla, Spain, 2005, pp. 2996–3000.

[4] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.

[5] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for

distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[6] S.-Y. Tu and A. H. Sayed, "Optimal combination rules for adaptation and learning over networks," in *Proc. IEEE CAMSAP*, San Juan, Puerto Rico, Dec. 2011, pp. 317–320.

[7] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over LMS adaptive networks," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5107–5124, Oct. 2012.

[8] C.-K. Yu and A. H. Sayed, "A strategy for adjusting combination weights over adaptive networks," in *Proc. IEEE ICASSP*, Vancouver, Canada, May 2013, pp. 4579–4583.

[9] T. C. Aysal, B. N. Oreshkin, and M. J. Coates, "Accelerated distributed average consensus via localized node state prediction," *IEEE Trans. Signal Proces.*, vol. 57, no. 4, pp. 1563–1576, 2009.

[10] V. Schwarz and G. Matz, "Nonlinear average consensus based on weight morphing," in *Proc. IEEE ICASSP*, Kyoto, Japan, 2012, pp. 3129–3132.

[11] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 60, pp. 6217–6234, Dec. 2012.

[12] J. Fernandez-Bes, J.A. Azpicueta-Ruiz, M.T.M. Silva, and J. Arenas-Garcia, "A novel scheme for diffusion networks with least-squares adaptive combiners," in *Proc. IEEE Int. Workshop Machine Learning for Signal Process*, Santander (Spain), Sep 2012, pp. 1–6.

[13] R. A Horn and C. R Johnson, *Matrix Analysis*, Cambridge University Press, 2012.

[14] G. H. Golub and C. F. C. F. Van Loan, *Matrix Computations*, JHU Press, 1996.

[15] A. H. Sayed, *Adaptive Filters*, Wiley, NJ, 2008.