Reduced-Complexity Distributed Least-Squares Estimation Over Adaptive Networks

Reza Arablouei and Kutluyl Doğançay
Institute for Telecommunications Research
University of South Australia
Mawson Lakes SA, Australia

Stefan Werner
Department of Signal Processing and Acoustics
School of Electrical Engineering, Aalto University
Espoo, Finland

Abstract—In wireless ad-hoc networks, nodes usually possess limited processing and electrical power resources. Therefore, when performing a decentralized task over such a network, it is desirable to minimize the required in-node computations and inter-node communications. In this paper, we propose a reduced-complexity diffusion recursive least-squares (RC-diffRLS) algorithm for distributed estimation over adaptive networks. To this end, we utilize the dichotomous coordinate-descent (DCD) algorithm to solve the normal equations of the local least-squares estimation problems at the nodes. Simulation results testify that the proposed algorithm can perform very close to a previously proposed diffusion recursive least-squares (diffRLS) algorithm while being considerably simpler in computational complexity and appreciably more resilient in numerical stability. Using the proposed algorithm, one can also establish a trade-off between complexity and performance.

Keywords—adaptive networks; dichotomous coordinate-descent iterations; diffusion adaptation; distributed estimation; recursive least-squares

I. INTRODUCTION

An adaptive network is a group of spatially-scattered agents (nodes) that are capable of data-processing and learning. The nodes are linked to each other and cooperate by exchanging information to perform decentralized real-time information processing and optimization. The local interactions disseminate the information across the network. Consequently, the nodes are able to adapt to changing statistics of the data and topographical conditions of the network [1]-[4].

Distributed estimation over adaptive networks has been the subject of extensive research. Several strategies, which rely on local interactions and in-network processing, have been proposed for this purpose. Among them are the diffusion [5]-[11], incremental [12]-[15], and consensus strategies [16]-[21]. In the diffusion strategies, the nodes communicate with all their neighbors while in the incremental strategies, they only communicate with their neighbors that are along a pre-defined cyclic path. This path has to run through all the nodes in the network. Defining such a path is not generally straightforward, especially for large networks. Similar to the diffusion strategies, the consensus strategies rely on the fusion of intermediate estimates of multiple neighboring nodes. However, unlike the diffusion strategies, the consensus strategies are explicitly constrained to converge to the same estimate at each node, i.e., the consensus state. It has been shown that, for distributed estimation over adaptive networks, the diffusion-based algorithms outperform their consensus-based counterparts [22].

The diffusion recursive least-squares (diffRLS) algorithm [5] implements distributed estimation over adaptive networks and approaches the optimal least-squares solution with no need for transmitting or inverting any matrix. It is in fact a devolved version of the conventional recursive least-squares (RLS) algorithm [23] and adaptively seeks a distributed solution for the global least-squares estimation problem across a network. Its computational complexity is proportional to square of the number of unknown parameters. Similar to the RLS algorithm, diffRLS may also suffer from numerical instability attributable to accumulation of round-off errors in finite-precision implementations.

In this paper, we develop a diffusion-based distributed recursive least-squares algorithm utilizing the dichotomous coordinate-descent (DCD) iterations [24]. The DCD algorithm is a shift-and-add algorithm that solves a system of linear equations (SLE) with no multiplication operation. In this algorithm, the number of exercised iterations and the step-size resolution control a trade-off between accuracy and complexity [25]. We employ the DCD algorithm in solving the SLEs of the normal equations corresponding to the local least-squares estimation problems. The new algorithm may be viewed as a reduced-complexity implementation of the diffRLS algorithm. Its performance can be made arbitrary close to that of the diffRLS algorithm while having a computational complexity significantly lower than the one of diffRLS. The saving in computational complexity is more pronounced when the input vectors have shift structure. In this case, the proposed algorithm requires an order of magnitude less computations compared with diffRLS. The proposed algorithm is also immune to the numerical instability caused by round-off error buildup in finite-precision environments, which might plague diffRLS.

II. ALGORITHM DESCRIPTION

Let us consider a connected network with $K$ nodes in which the nodes aim to collectively identify an unknown parameter vector, $\mathbf{h} \in \mathbb{R}^{L \times 1}$. Each node observes an input vector, $\mathbf{x}_{k,n} \in \mathbb{R}^{L \times 1}$, and an output signal, $d_{k,n} \in \mathbb{R}$, that arise from a linear system described by

This work was supported in part by a Commonwealth Scientific and Industrial Research Organisation (CSIRO) scholarship, the Academy of Finland, and the Centre of Excellence in Smart Radios and Wireless Research (SMARAD) at the Aalto University.
\[ d_{k,n} = x_{k,n}^T h + v_{k,n}. \]

Here, \( k \in \{1, 2, ..., K\} \) is the node index and \( n \in \mathbb{N} \) is the time index. Superscript \( T \) denotes matrix/vector transposition and \( v_{k,n} \in \mathbb{R} \) represents the background noise.

Nodes can iteratively estimate the unknown parameter vector, \( h \), exploiting the data available up to the current time within their neighborhoods. This can be realized in the following four steps.

1) Each node transmits its input-output data to its neighbors.

2) Each node computes an intermediate estimate, \( z_{k,n} \in \mathbb{R}^{l 	imes 1} \), by minimizing its local exponentially-weighted least-squares cost function using the input-output data available within its neighborhood (adaptation):

\[ z_{k,n} = \arg\min_z \sum_{l \in N_k} c_{l,k} \| d_{l,n} - x_{l,n}^T z \|^2 \]  \hspace{1cm} (1)

where

\[ d_{k,n} = \left[ d_{k,n}, \lambda^{1/2} d_{k,n-1}, ..., \lambda^{(n-1)/2} d_{k,1} \right]^T, \]

\[ X_{k,n} = \left[ x_{k,n}, \lambda^{1/2} x_{k,n-1}, ..., \lambda^{(n-1)/2} x_{k,1} \right], \]

\( \| \cdot \| \) denotes the Euclidean norm, and \( 0 < \lambda < 1 \) is a forgetting factor.

3) Each node transmits its intermediate estimate vector to its neighbors.

4) Each node makes an estimate, \( w_{k,n} \in \mathbb{R}^{l 	imes 1} \), by weighted-averaging the intermediate estimates available within its neighborhood (combination):

\[ w_{k,n} = \sum_{l \in N_k} a_{l,k} z_{l,n}. \]  \hspace{1cm} (2)

The set \( N_k \) denotes the closed neighborhood of node \( k \). It contains \( k \) and indexes of the nodes that are directly connected to node \( k \) and can exchange information with it. Moreover, the coefficients \( c_{l,k} > 0 \) and \( a_{l,k} > 0 \) are used to assign different weights to the data coming from different nodes based on their reliability or importance. They satisfy

\[ c_{l,k} = a_{l,k} = 0 \text{ if } l \notin N_k \text{ } \forall k, \]

\[ 1^T C = 1^T, \]

\[ C 1 = 1, \]

and

\[ 1^T A = 1^T \]

where \( c_{l,k} \) and \( a_{l,k} \) are the \((l,k)\)th entries of \( C \in \mathbb{R}^{K \times K} \) and \( A \in \mathbb{R}^{K \times K} \), respectively, and \( 1 \in \mathbb{R}^{K \times 1} \) is the all-ones vector.

The solution of the optimization problem (1) is the solution of the following normal equations:

\[ \left( \sum_{l \in N_k} c_{l,k} X_{l,n} X_{l,n}^T \right) z_{k,n} = \sum_{l \in N_k} c_{l,k} X_{l,n} d_{l,n} \]

or

\[ R_{k,n} z_{k,n} = p_{k,n} \]  \hspace{1cm} (3)

where

\[ R_{k,n} = \sum_{l \in N_k} c_{l,k} X_{l,n} X_{l,n}^T \]

\[ = \lambda R_{k,n-1} + \sum_{l \in N_k} c_{l,k} X_{l,n} X_{l,n}^T \]

\( = \lambda R_{k,n-1} + \sum_{l \in N_k} c_{l,k} X_{l,n} X_{l,n}^T \]  \hspace{1cm} (4)

is the exponentially-weighted input autocorrelation matrix at node \( k \) and

\[ p_{k,n} = \sum_{l \in N_k} c_{l,k} X_{l,n} d_{l,n} \]

\[ = \sum_{l \in N_k} \left( c_{l,k} \sum_{i=1}^n \lambda^{n-i} x_{l,i} x_{l,i}^T \right) \]

\[ = \lambda p_{k,n-1} + \sum_{l \in N_k} c_{l,k} X_{l,n} d_{l,n} \]  \hspace{1cm} (5)

is the exponentially-weighted cross-correlation vector between the input vector and the output signal at node \( k \).

The diffusion recursive least-squares (diffRLS) algorithm [5] recursively calculates

\[ z_{k,n} = R_{k,n}^{-1} p_{k,n} \]

by applying the matrix inversion lemma [26] to (4) for \( n_k \) times and replacing \( z_{k,n-1} \) with \( w_{k,n-1} \) in the resultant recursions for \( z_{k,n} \). Using \( w_{k,n-1} \) rather than \( z_{k,n-1} \) allows the local estimates, \( w_{k,n} \), to diffuse outside of each node’s own neighborhood. Here, \( n_k \) denotes the cardinality of \( N_k \).

As an alternative approach, the system of linear equations (SLE) in (3) can be efficiently solved utilizing the dichotomous coordinate-descent (DCD) algorithm [24]. However, in order to implement diffusion adaptation, instead of (3), we solve

\[ R_{k,n} v_{k,n} = b_{k,n} \]  \hspace{1cm} (6)

where

\[ v_{k,n} = z_{k,n} - w_{k,n-1} \]

and

\[ b_{k,n} = p_{k,n} - R_{k,n} w_{k,n-1}. \]  \hspace{1cm} (7)

Substituting (4) and (5) into (7) results in

\[ b_{k,n} = \lambda p_{k,n-1} + \sum_{l \in N_k} c_{l,k} X_{l,n} d_{l,n} - \lambda R_{k,n-1} w_{k,n-1} \]

\[ - \sum_{l \in N_k} c_{l,k} X_{l,n} X_{l,n}^T w_{k,n-1} \]

\[ = \lambda r_{k,n-1} + \sum_{l \in N_k} c_{l,k} X_{l,n} (d_{l,n} - X_{l,n}^T w_{k,n-1}) \]

where
It is known that by partial updating, one can trade performance for complexity. Three design parameters, $N \in \mathbb{N}$, $M \in \mathbb{N}$, and $H \in \mathbb{R}$ govern the accuracy and complexity of the DCD algorithm [24]. The entries of the solution vector, $\mathbf{v}_k$, are represented as fixed-point words with $M$ bits within an amplitude range of $[-H, H]$. The algorithm performs maximum $N$ iterative updates, i.e., $N$ defines the maximum number of entries in $\mathbf{z}_k$ that are updated at each time instant. Therefore, when $N < L$, the algorithm realizes a form of selective partial updates [27]. It is known that by partial updating, one can trade performance for complexity [28].

### III. Computational Complexity

Since $\mathbf{R}_{k,n} \forall k, n$ is symmetric, we only need to compute its upper triangular part. If we choose the forgetting factor as $\lambda = 1 - 2^{-s}$ with $s$ being a positive integer, multiplications by $\lambda$ can be replaced by bit-shifts and additions [25]. Moreover, when

$$\mathbf{x}_{k,n} = [x_{k,0}, x_{k,n-1}, \ldots, x_{k,n-L+1}]^T$$

where $x_{k,n} \in \mathbb{R}$ is the input signal, i.e., there is shift structure in the input data, updating $\mathbf{R}_{k,n}$ is considerably simplified. In this case, the upper-left $(L - 1) \times (L - 1)$ block of $\mathbf{R}_{k,n-1}$ can be copied to the lower-right $(L - 1) \times (L - 1)$ block of $\mathbf{R}_{k,n}$. Hence, only the first row and the first column of $\mathbf{R}_{k,n}$ need be updated directly. Due to the symmetry of $\mathbf{R}_{k,n}$, it is sufficient to update only the first column via

$$\mathbf{R}_{k,n}^{(1)} = \lambda \mathbf{R}_{k,n-1}^{(1)} + \mathbf{x}_{k,n} \mathbf{x}_{k,n}^T.$$

In Table III, we present the number of arithmetic operations required by the diffRLS and RC-diffRLS algorithms at each node and each iteration for both shift-structured and non-shift-structured input cases.

### IV. Simulations

Consider a problem of distributed system identification over an adaptive network. The unknown system has $L = 8$ random parameters and unit energy. The network consists of $K = 20$ nodes. Each node is connected on average to three other nodes. The input vectors at the nodes are shift-structured and the input signals, $\mathbf{x}_{k,n}$, are zero-mean Gaussian and uncorrelated in time and space. The additive noises at the nodes, $\mathbf{v}_{k,n}$, are also zero-mean Gaussian. The variances of the input signals and the noises at each node are shown in Fig. 1.
we use metropolis weights [29] for $\mathbf{C}$ (at the adaptation step) and simple averaging (uniform) weights [30] for $\mathbf{A}$ (at the combination step). We also initialize the estimates to all-zero vectors and set $\lambda = 1 - 2^{-7} = 0.9922$, $M = 12$, and $H = 1$.

In Figs. 2 and 3, we compare the performance of the diffRLS algorithm and the RC-diffRLS algorithm with different values of $N$. Fig. 2 shows the misalignment curves averaged over all the nodes. The misalignment is defined as

$$E \left[ ||\mathbf{h} - \mathbf{w}_{k,n}||^2 \right]$$

and evaluated by ensemble-averaging over $10^3$ independent runs. Fig. 3 shows the node-specific steady-state misalignments that are obtained by averaging over 100 steady-state values. It is clear that the larger $N$ is, the closer to diffRLS the performance of RC-diffRLS is. However, exercising only a single iteration of the DCD algorithm, i.e., $N = 1$, still results in RC-diffRLS performing very close to diffRLS. Specifically, there is less than 1 dB difference in the misalignments of the two algorithms.

To examine the numerical stability of the RC-diffRLS algorithm, in Fig. 4, we plot the instantaneous misalignment, i.e., $||\mathbf{h} - \mathbf{w}_{k,n}||^2$, of the diffRLS and RC-diffRLS algorithms for $10^4$ iterations and $N = 12$. It is seen that, unlike diffRLS that diverges after about 4200 iterations, RC-diffRLS maintains its numerical stability throughout the simulation. In addition, we carried out several numerical experiments in different scenarios for very large numbers of iterations, e.g., $10^5$. Consistent with the findings of [25] pertaining to the numerical stability of the DCD-RLS algorithm, during the experiments, we did not observe any numerical instability in the RC-diffRLS algorithm. Moreover, although not shown here, similar results in terms of misalignment performance and numerical stability are obtained in the absence of shift structure in the input vectors.

In Table IV, we present the average number of required arithmetic operations by the diffRLS and RC-diffRLS algorithms at each node and each iteration when

| $N$ | diffRLS | RC-diffRLS, $N = 1$ | RC-diffRLS, $N = 4$ | RC-diffRLS, $N = 8$ | RC-diffRLS, $N = 12$ |
|-----|---------|------------------|-----------------|-----------------|-----------------
|     |         |                  |                 |                 |                  |
| 1   |         |                  |                 |                 |                  |
| 4   |         |                  |                 |                 |                  |
| 8   |         |                  |                 |                 |                  |
| 12  |         |                  |                 |                 |                  |

In Table IV, we present the average number of required arithmetic operations by the diffRLS and RC-diffRLS algorithms at each node and each iteration when $N = 12$ in the...
above experiment. The numbers are given for both cases of shift-structured and non-shift-structured inputs.

V. CONCLUSION

We proposed a reduced-complexity diffusion recursive least-square algorithm by employing the dichotomous coordinate-descent (DCD) iterations. We utilized the DCD algorithm to solve the node-specific normal equations associated with the least-squares estimation over a diffusive adaptive network. Simulation results confirmed that the proposed algorithm could perform as well as a previously proposed diffusion recursive least-square (diffRLS) algorithm with the same communication complexity but significantly lower computational complexity and improved numerical stability in addition to providing a trade-off between complexity and performance.

REFERENCES


