Diffusion Spline Adaptive Filtering

Simone Scardapane, Michele Scarpiniti, Danilo Comminiello, and Aurelio Uncini
Department of Information Engineering, Electronics and Telecommunications (DIET)
"Sapienza" University of Rome
Email: simone.scardapane@uniroma1.it

***Abstract*—Diffusion adaptation (DA) algorithms allow a network of agents to collectively estimate a parameter vector, by jointly minimizing the sum of their local cost functions. This is achieved by interleaving local update steps with 'diffusion' steps, where information is combined with their own neighbors. In this paper, we propose a novel class of nonlinear diffusion filters, based on the recently proposed spline adaptive filter (SAF). A SAF learns nonlinear models by local interpolating polynomials, with a small overhead with respect to linear filters. This arises from the fact that only a small subset of parameters of the nonlinear component are adapted at every time-instant. By applying ideas from the DA framework, in this paper we derive a diffused version of the SAF, denoted as D-SAF. Experimental evaluations show that the D-SAF is able to robustly learn the underlying nonlinear model, with a significant gain compared to a non-cooperative solution.**

I. Introduction

Lately, there has been a renewed interest in the problems of decentralized inference and estimation, topics with a long history in the signal processing community [1]. With respect to standard linear filtering, a particularly promising line of research can be found in the field of diffusion filtering (DF) [2]. DF algorithms were originally proposed to extend linear filtering to the distributed case, by alternating local update steps (by means of classical gradient descent procedures) with global 'diffusion' steps, where each node combines the information provided by its own neighbors. By considering only localized forms of communication, thus, they are able to be deployed in a wide range of situations, with strong capabilities of scaling up to very large networks. Originally proposed for adaptation using LMS-like updates [3], named the diffusion LMS (D-LMS), most of the standard linear filtering algorithms have now been mapped to the DF field and have found several successful applications [2], [4].

Clearly, their applicability is limited to scenarios where the assumption of a linear model between the output and the observed variables is meaningful. In case where a nonlinear model is needed, several alternatives (or extensions) of DF algorithms exist. Basically, they can be categorized in three classes, including (i) distributed versions of kernel methods [5], [6]; (ii) linear filters combined with a fixed nonlinear expansion [7], [8]; and (iii) algorithms framed in the distributed optimization and machine learning frameworks [9]–[11]. This last class includes extensions of the DF theory to the distributed optimization scenario, which is denoted as Diffusion Adaptation (DA) [2]. However, each of these approaches typically loose one or more of the features of linear DF algorithms, either in terms of easiness of implementation, requirements on the topology, or real-time constraints.

In this paper, we propose a novel class of nonlinear DF algorithms based on the recently proposed spline adaptive filter (SAF) [12]. SAFs are attractive nonlinear filters for two main reasons. First, the nonlinear part is linear-in-the-parameters (LIP), allowing for the possibility of adapting both parts of the filter using standard linear filtering techniques. Secondly, while the spline is defined by a potentially large number of parameters, only a small subset of them is considered and adapted at each time step (4 in our experiments). Due to this, they allow to approximate nontrivial nonlinear functions with a small increase in complexity with respect to linear filters.

As an initial step towards distributed SAFs, in this paper we focus on the Wiener SAF [12], where a linear filter is followed by an adaptive nonlinear transformation, obtained with spline interpolation. Based on the general theory of DF [11], we propose a diffused version denoted as D-SAF. In particular, we show that a cooperative behavior can be implemented by considering two subsequent diffusion operations, on the linear and nonlinear components of the SAF respectively. Due to this, the D-SAF inherits the aforementioned characteristics of the centralized SAF, namely, it enables the agents to collectively estimate a nonlinear function, with a small overhead with respect to a purely linear DF. In fact, D-LMS can be shown to be a special case of D-SAF, where adaptation is restricted to the linear part only. Simulations show that the D-SAF is able to efficiently learn the underlying model, and outperforms D-LMS and a non-cooperative SAF.

The rest of the paper is organized as follows. In Section II we introduce the basic framework of spline interpolation and SAFs. Section III formulates the D-SAF algorithm. Subsequently, we detail our experimental results in Section IV. Finally, Section V presents some concluding remarks.

II. Spline Adaptive Filter

Denote by $x[n]$ the input to the SAF filter at time n, and by $\mathbf{x}_n = [x[n], \dots, x[n-M+1]]^T$ a buffer of the last M samples. In this paper, we assume to be dealing with real inputs, and that an unknown Wiener model is generating the desired response as follows:

$$d[n] = f_0\left(\mathbf{w}_0^T \mathbf{x}_n\right) + \nu[n]\,, \tag{1}$$

where $\mathbf{w}_0 \in \mathbb{R}^M$ are the linear coefficients, $f_0(\cdot)$ is a desired nonlinear function, which is supposed continuous and differentiable, and $\nu[n] \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian noise term.

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

Similarly, a SAF computes the output in a two-step fashion. First, it performs a linear filtering operation given by:

$$s[n] = \mathbf{w}_n^T \mathbf{x}_n \,. \tag{2}$$

Then, the final output is computed via spline interpolation over $s[n]$ [12], [13]. A spline is a flexible polynomial defined by a set of Q control points (called *knots*), and denoted as $\mathbf{Q}_i = [q_{x,i}\ q_{y,i}]$. We suppose that the knots are uniformly distributed, i.e. $q_{x,i+1} = q_{x,i} + \Delta x$, for a fixed $\Delta x \in \mathbb{R}$. Without lack of generality, we also constrain the knots to be symmetrically spaced around the origin. Given the output of the linear filter $s[n]$, the spline is defined as an interpolating polynomial of order P, passing by the closest knot to $s[n]$ and its P successive knots. In particular, given the index i of the closest knot, we can define the normalized abscissa value between $q_{x,i}$ and $q_{x,i+1}$ as:

$$u = \frac{s[n]}{\Delta x} - \left\lfloor \frac{s[n]}{\Delta x} \right\rfloor \,. \tag{3}$$

From u we can compute the normalized reference vector $\mathbf{u} = \begin{bmatrix} u^P\ u^{P-1} \dots u\ 1 \end{bmatrix}^T$, while from i we can extract the relevant control points $\mathbf{q}_{i,n} = [q_{y,i}\ q_{y,i+1} \dots q_{y,i+P}]^T$. We refer to the vector $\mathbf{q}_{i,n}$ as the ith *span*. The output of the filter is:

$$y[n] = \varphi(s[n]) = \mathbf{u}^T \mathbf{B} \mathbf{q}_{i,n} \,, \tag{4}$$

where $\varphi(s[n])$is the adaptable nonlinearity, and $\mathbf{B} \in \mathbb{R}^{(P+1)\times(P+1)}$ is called the spline basis matrix. In this work, we use the Catmull-Rom (CR) spline with $P = 3$, given by:

$$\mathbf{B} = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \,. \tag{5}$$

Several alternative choices are available, such as the B-spline matrix [12]. Both (2) and (4) are LIP, and can be adapted with the use of any standard linear filtering technique. Applying the chain rule, it is straightforward to compute the derivative of the SAF output with respect to the linear coefficients:

$$\begin{aligned} \frac{\partial \varphi(s[n])}{\partial \mathbf{w}_n} &= \frac{\partial \varphi(s[n])}{\partial u} \cdot \frac{\partial u}{\partial s[n]} \cdot \frac{\partial s[n]}{\partial \mathbf{w}_n} = \\ &= \dot{\mathbf{u}} \mathbf{B} \mathbf{q}_{i,n} \left(\frac{1}{\Delta x} \right) \mathbf{x}_n \,, \end{aligned} \tag{6}$$

where:

$$\dot{\mathbf{u}} = \frac{\partial \mathbf{u}}{\partial u} = \begin{bmatrix} P u^{P-1}\ (P-1) u^{P-2} \dots 1\ 0 \end{bmatrix}^T \,. \tag{7}$$

Similarly, for the nonlinear part we obtain:

$$\frac{\partial \varphi(s[n])}{\partial \mathbf{q}_{i,n}} = \mathbf{B}^T \mathbf{u} \,. \tag{8}$$

In this paper, we consider a first-order adaptation for both the linear and the nonlinear part of the SAF. Defining the error $e[n] = d[n] - y[n]$, we aim at minimizing the expected mean-squared error given by:

$$J(\mathbf{w}, \mathbf{q}) = \mathrm{E}\left\{ e[n]^2 \right\} \,, \tag{9}$$

where $\mathbf{q} = [q_{y,1}, \dots, q_{y,Q}]^T$. As is standard approach, we approximate (9) with the instantaneous error given by $\hat{J}(\mathbf{w}, \mathbf{q}) = e[n]^2$, then, we apply two simultaneous steepest-descent steps to solve the overall optimization problem:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu_w e[n] \varphi'(s[n]) \mathbf{x}_n \,, \tag{10}$$

$$\mathbf{q}_{i,n+1} = \mathbf{q}_{i,n} + \mu_q e[n] \mathbf{B}^T \mathbf{u} \,, \tag{11}$$

where we defined $\varphi'(s[n]) = \dot{\mathbf{u}} \mathbf{B} \mathbf{q}_{i,n} \left(\frac{1}{\Delta x} \right)$, and we use two possibly different step-sizes $\mu_w, \mu_q > 0$. Note that in (11) we adapt only the coefficients related to the ith span, since it can easily be shown that $\frac{\partial \hat{J}(\mathbf{w}_n, \mathbf{q}_n)}{\partial \mathbf{q}_n}$ is 0 for all the coefficients outside the span. Coefficients of the spline are initialized such that $\varphi(s[n]) = s[n]$. Using this initialization criterion, the LMS filter can be considered as a special case of the SAF, where adaptation is restricted to the linear part, i.e. $\mu_q = 0$.

III. Diffusion SAF

In the rest of the paper, we employ the standard network model as in the DF literature [2]. More in particular, we consider a network of L agents (or nodes), indexed by the integers $1, \dots, L$. The connectivity of the network is represented in its entirety by a real-valued matrix $\mathbf{C} \in \mathbb{R}^{L \times L}$, where entry $C_{kl} > 0$ if nodes k and l are connected, 0 otherwise. We assume that communication between two nodes k and l is allowed only if $C_{kl} > 0$, i.e., only if k is in the immediate neighborhood of l. The symbol $\mathcal{N}_k$ will denote the inclusive neighborhood of node k. Generally speaking, the matrix $\mathbf{C}$ is used by the agents to fuse information coming from their neighborhoods. For this reason, we refer to $\mathbf{C}$ as the mixing matrix, and to a single entry C_{kl} as a mixing coefficient. As is standard assumption, we require the mixing coefficients to define a convex combination for every node:

$$C_{kl} \geq 0 \text{ and } \sum_{l=1}^{L} C_{kl} = 1 \quad k, l = 1, \dots, L \,. \tag{12}$$

Formally, any left-stochastic matrix is a valid mixing matrix [2]. In our experiment, we consider the well-known Metropolis weights [14].

At a generic time instant n, each agent receives some input/output data denoted by $\left(\mathbf{x}_n^{(k)}, d^{(k)}[n] \right)$, where we introduce an additional superscript (k) for explicating the node dependence. We assume that streaming data at the local level is generated similarly to (1), according to:

$$d^{(k)}[n] = f_0 \left(\mathbf{w}_0^T \mathbf{x}_n^{(k)} \right) + \nu^{(k)}[n] \,. \tag{13}$$

More in particular, we assume that $\mathbf{w}_0$ and $f_0(\cdot)$ are shared over the network, which is a reasonable assumption in many situations [2]. Each node, however, receives input data with possibly different autocorrelation $\mathbf{R}_u^{(k)} = \mathrm{E}\left\{ \mathbf{x}^{(k)T} \mathbf{x}^{(k)} \right\}$, and different additive noise terms $\nu^{(k)}[n] \sim \mathcal{N}(0, \sigma_k^2)$. Additionally, we assume that the nodes have agreed beforehand

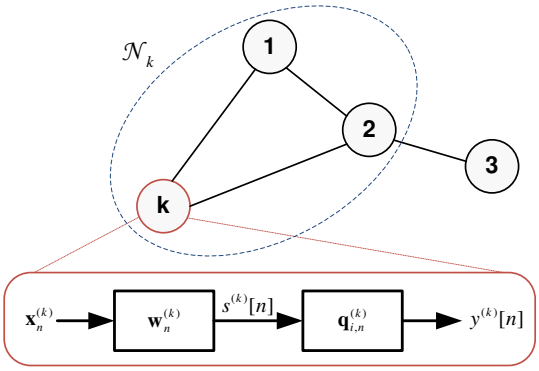


Fig. 1. Schematic depiction of SAF interpolation performed over a network of agents. Each agent is connected to a neighborhood of other agents, and at every time instant it updates a local estimate of the optimal SAF model.

on a specific spline basis matrix $\mathbf{B}$, and on a set of initial control points $\mathbf{q}_0$. Both quantities are common throughout the network. This is shown schematically in Fig. 1.

Given these assumptions, the network objective is to find the optimal SAF parameters $(\mathbf{w}, \mathbf{q})$ such that the following global cost function is minimized:

$$J_{\text{glob}}(\mathbf{w}, \mathbf{q}) = \sum_{k=1}^{L} J_{\text{loc}}^{(k)}(\mathbf{w}, \mathbf{q}) = \sum_{k=1}^{L} \mathrm{E}\left\{e^{(k)}[n]^2\right\}, \quad (14)$$

where each expectation is defined with respect to the local input statistics. To solve this, we consider an approach inspired to the ideas of DF algorithms [2]. As explained in Section I, the main idea of DF techniques is to interleave parallel adaptation steps with diffusion steps, where information on the current estimates are locally combined based on the mixing matrix $\mathbf{C}$ (see for example [2, Section V-B]). Denote by $\left(\mathbf{w}_n^{(k)}, \mathbf{q}_n^{(k)}\right)$ the SAF estimate of node k at time-instant n. In the diffusion SAF (D-SAF), each node starts by diffusing its own estimate of the linear part of the SAF filter:

$$\boldsymbol{\psi}_n^{(k)} = \sum_{l \in \mathcal{N}_k} C_{kl} \mathbf{w}_n^{(l)}. \quad (15)$$
w-diffusion

Next, we can use the new weights $\boldsymbol{\psi}_n^{(k)}$ to compute the linear output of the filter as $s^{(k)}[n] = \boldsymbol{\psi}_n^{(k)T} \mathbf{x}_n^{(k)}$. From this, each node can identify its current span index i. In the second phase, the nodes perform a second diffusion step over their span:

$$\boldsymbol{\xi}_{i,n}^{(k)} = \sum_{l \in \mathcal{N}_k} C_{kl} \mathbf{q}_{i,n}^{(l)}. \quad (16)$$
q-diffusion

The q-diffusion step requires combination only of the coefficients in the span $\mathbf{q}_{i,n}^{(k)}$, hence its complexity is independent of the number of control points in the spline, being defined only by the spline order P. Due to this, it also requires significantly less communication than the w-diffusion step: each node sends its span index i to the neighbors, receiving back the vectors $\mathbf{q}_{i,n}^{(l)}$. For simplicity, the mixing weights C_{kl} in the two diffusion steps are assumed identical.

Once the nodes have diffused their information, they can proceed to a standard adaptation step as in the single-agent case. In particular, the spline output given the new span is obtained as:

$$y^{(k)}[n] = \varphi_k(s^{(k)}[n]) = \mathbf{u}^T \mathbf{B} \boldsymbol{\xi}_{i,n}^{(k)}. \quad (17)$$

From this, the local error is given as $e^{(k)}[n] = d^{(k)}[n] - y^{(k)}[n]$. The two gradient descent steps are then:

$$\mathbf{w}_{n+1}^{(k)} = \boldsymbol{\psi}_n^{(k)} + \mu_w^{(k)} e^{(k)}[n] \varphi'(s^{(k)}[n]) \mathbf{x}_n^{(k)}, \quad (18)$$
w-adapt

$$\mathbf{q}_{i,n+1}^{(k)} = \boldsymbol{\xi}_{i,n}^{(k)} + \mu_k^{(k)} e^{(k)}[n] \mathbf{B}^T \mathbf{u}. \quad (19)$$
q-adapt

where the two step sizes $\mu_w^{(k)}, \mu_q^{(k)}$ are possibly different across different agents. The overall algorithm is summarized in Algorithm I. Note that in the current paper we consider a diffusion step prior to the adaptation step. In the DF literature, this is known as a combine-then-adapt (CTA) strategy. This is true even if the two diffusion steps are not consecutive in Algorithm I. In fact, Algorithm I is equivalent to the case where the full vector $\mathbf{q}_n^{(k)}$ is exchanged before selecting the proper span. Following similar reasonings, we can easily obtain an adapt-then-combine (ATC) strategy by inverting the two steps. Additionally, similarly to what we remarked in Section II, we note that D-LMS [3] is a special case of the D-SAF, where each node initializes its nonlinearity as the identity, and $\mu_q^{(k)} = 0, k = 1, \ldots, L$.

Algorithm I
SUMMARY OF THE D-SAF ALGORITHM (CTA VERSION).

Initialize $\mathbf{w}_{-1}^{(k)} = \delta[n], \mathbf{q}_0^{(k)}$, for $k = 1, \ldots, L$
for $n = 0, 1, \ldots$ **do**
  for $k = 1, \ldots, L$ **do**
    $\boldsymbol{\psi}_n^{(k)} = \sum_{l \in \mathcal{N}_k} C_{kl} \mathbf{w}_n^{(k)}$
    $s^{(k)}[n] = \boldsymbol{\psi}_n^{(k)T} \mathbf{x}_n^{(k)}$
    $u = s^{(k)}[n]/\Delta x - \lfloor s^{(k)}[n]/\Delta x \rfloor$
    $i = \lfloor s^{(k)}[n]/\Delta x \rfloor + (Q-1)/2$
    $\boldsymbol{\xi}_{i,n}^{(k)} = \sum_{l \in \mathcal{N}_k} C_{kl} \mathbf{q}_{i,n}^{(k)}$
    $y^{(k)}[n] = \mathbf{u}^T \mathbf{B} \boldsymbol{\xi}_{i,n}^{(k)}$
    $e^{(k)}[n] = d^{(k)}[n] - y^{(k)}[n]$
    $\mathbf{w}_{n+1}^{(k)} = \boldsymbol{\psi}_n^{(k)} + \mu_w^{(k)} e^{(k)}[n] \varphi'(s^{(k)}[n]) \mathbf{x}_n^{(k)}$
    $\mathbf{q}_{i,n+1}^{(k)} = \boldsymbol{\xi}_{i,n}^{(k)} + \mu_k^{(k)} e^{(k)}[n] \mathbf{B}^T \mathbf{u}$
  end for
end for

IV. EXPERIMENTAL SETUP

To test the proposed D-SAF, we consider a network topology with $L = 10$ agents, whose connectivity is generated randomly, such that every pair of nodes has a 60% probability of being connected. Data is generated according to the Wiener model in (13), where the optimal weights $\mathbf{w}_0$ are extracted

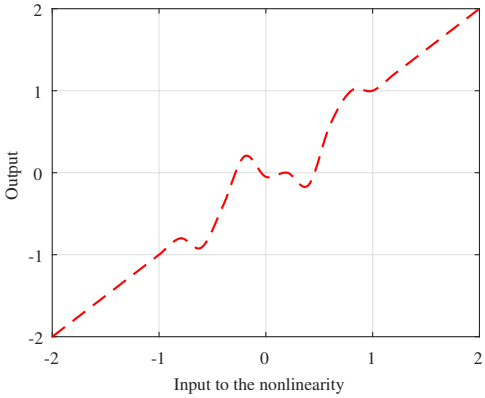


Fig. 2. Nonlinear distortion applied to the output signal.

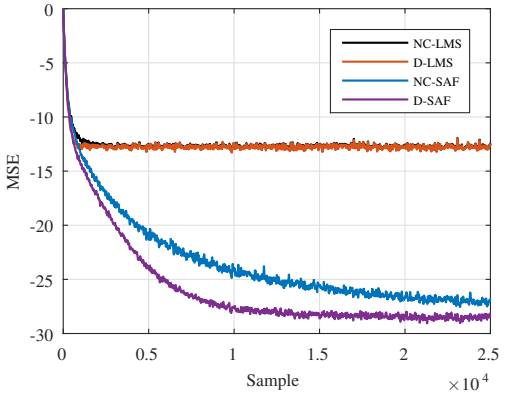


Fig. 3. MSE evolution, averaged across the nodes.

randomly from a normal distribution, while the nonlinearity $f_0(\cdot)$ for the experiment is depicted in Fig. 2. This represents a mild nonlinearity. The input signal at each node is generated following the experiments in [12], and it consists of 25000 samples generated according to:

$$x_k[n] = a_k x_k[n-1] + \sqrt{1-a_k^2}\epsilon[n]\,, \tag{20}$$

where the correlation coefficients a_k are assigned randomly at every node from an uniform probability distribution in $[0, 0.8]$, while $\epsilon[n]$ is a white Guassian noise term with zero mean and unitary variance. The desired signal is then given by (13), where the noise variances $\sigma^{(k)}[n]$ at every node are assigned randomly in $[-10, -25]$ dB. In all experiments, knots are equispaced in $[-2, +2]$ with $\Delta x = 0.2$.

We compare D-SAF with a non-cooperative SAF (denoted as NC-SAF), which corresponds in choosing a diagonal mixing matrix $\mathbf{C} = \mathbf{I}$. Similarly, we compare with the standard D-LMS [3], and a non-cooperative LMS, denoted as NC-LMS. Experiments are repeated 15 times, by keeping fixed the topology of the network and the optimal parameters of the system. Results are then averaged throughout the nodes. MATLAB code is available under open-source license.[1]

Local correlation coefficients, noise variances and local step-sizes are chosen randomly in the intervals $[0, 0.7]$, $\left[0, 10^{-2}\right]$ and $\left[0, 10^{-2}\right]$ respectively. These settings allow a certain amount of variety on the network. The first measure of error that we consider is the mean-squared error (MSE), defined in dB as:

$$\text{MSE}_k[n] = 10\log\left(\text{E}\left[(d_k[n] - y_k[n])^2\right]\right)\,. \tag{21}$$

Results in term of MSE are given in Fig. 3, where the proposed algorithm is shown with a solid violet line. The MSE is computed by averaging (21) over the different nodes. As expected, due to the nonlinear distortion, LMS achieves a generally poor performance, with a steady-state MSE of -12 dB. Additionally, there is almost no improvement when considering D-LMS compared to NC-LMS. The SAF filters are instead able to approximate extremely well the desired system. The diffusion strategy, however, provides a significant improvement in convergence time with respect to the non-cooperative version, as is evident from Fig. 3. Further clarifications on the two algorithms can be obtained by considering the linear mean-squared deviation (MSD):

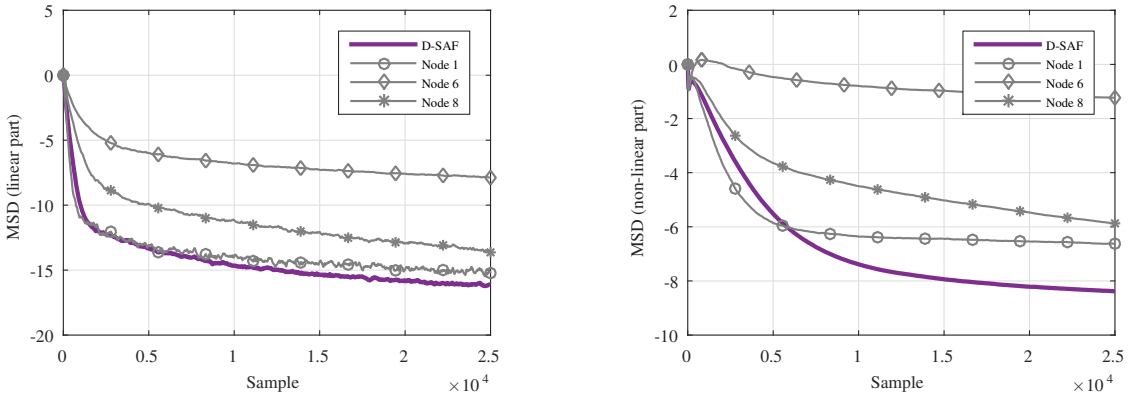
$$\text{MSD}_k^{\text{l}} = 10\log\left(\text{E}\left[\|\mathbf{w}_0 - \mathbf{w}_n^{(k)}\|_2^2\right]\right)\,, \tag{22}$$

and similarly for the nonlinear MSD. The overall behavior of the MSD is shown in Fig. 4. In particular, we plot the MSD evolution for D-SAF and for three representative agents running NC-SAF. It can be seen that, due to the differences in configuration, some nodes have a much slower convergence than other, such as node 6 compared to node 1. However, these statistical variations are successfully averaged out by the diffusion algorithm, which in the experiments was found to outperform even the fastest node in the network. The resulting nonlinear models for three representative nodes running NC-SAF, and for the nodes running D-SAF are shown in Fig. 5.

V. Conclusions

DA allows to derive distributed optimization protocols, that only rely on limited (local) exchange of information between agents of a network. In this paper, we have introduced a distributed algorithm for adapting a particular class of nonlinear filters, called SAF, using the general framework of DF. The algorithm inherits the properties of SAFs in the centralized case, namely, it allows for a flexible nonlinear estimation of the underlying function, with a relatively small increase in computational complexity. In particular, the algorithm can be implemented with two diffusion steps, and two gradient descent steps, thus requiring in average only twice as much computations as the standard D-LMS. Our experimental results show that D-SAF is able to efficiently learn hard nonlinearities, with a definite increase in convergence time with respect to a non-cooperative implementation. In this work, we have focused on a first-order adaptation algorithm, with CTA combiners. In future works, apart from a thorough theoretical analysis of its convergence properties, we plan to extend the D-SAF algorithm to the case of second-order adaptation with Hessian

[1] https://bitbucket.org/ispamm/diffusion-spline-filtering



(a) Linear MSD (b) nonlinear MSD

Fig. 4. MSD evolution for D-SAF and 3 representative nodes running NC-SAF.

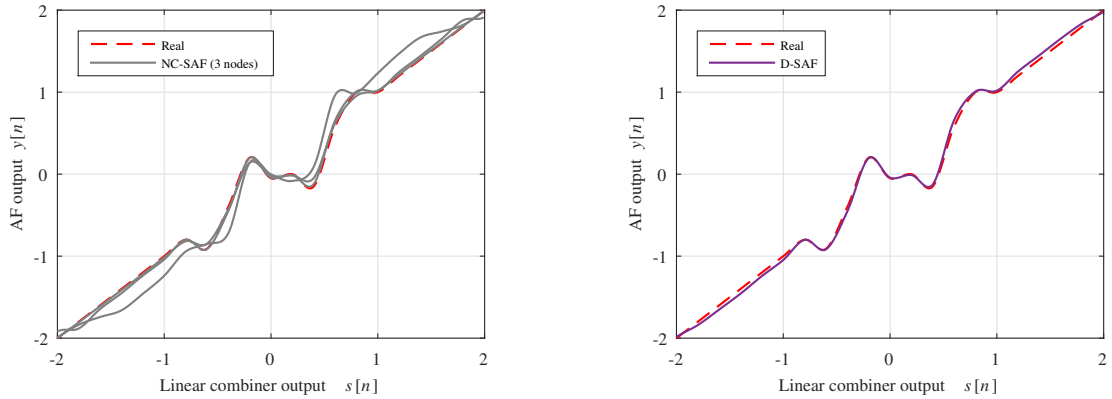


Fig. 5. Final estimation of the nonlinear model. (a) Three representative nodes running NC-SAF. (b) Final spline of the nodes running D-SAF.

information, ATC combiners [3], and asynchronous networks [15]. Additionally, we plan on investigating diffusion protocols for more general architectures, including Hammerstein [16] and IIR spline filters.

REFERENCES

[1] R. Viswanathan and P. K. Varshney, "Distributed detection with multiple sensors I. Fundamentals," *Proc. IEEE*, vol. 85, no. 1, pp. 54–63, 1997.

[2] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[3] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, 2008.

[4] P. Di Lorenzo and A. H. Sayed, "Sparse distributed learning based on diffusion adaptation," *IEEE Trans. Signal Process.*, vol. 61, no. 6, pp. 1419–1433, 2013.

[5] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, 2006.

[6] P. Honeine, M. Essoloh, C. Richard, and H. Snoussi, "Distributed regression in sensor networks with a reduced-order kernel model," in *Proc. of the 2008 IEEE Global Telecommunications Conference (GLOBECOM'08)*. New Orleans, LA, USA: IEEE, 2008, pp. 1–5.

[7] S. Scardapane, D. Wang, M. Panella, and A. Uncini, "Distributed learning for Random Vector Functional-Link networks," *Information Sciences*, vol. 301, pp. 271 – 284, 2015.

[8] S. Scardapane, D. Wang, and M. Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Networks*, vol. 78, pp. 65–74, 2016.

[9] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, 2010.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[11] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, 2012.

[12] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini, "Nonlinear spline adaptive filtering," *Signal Processing*, vol. 93, no. 4, pp. 772 – 783, 2013.

[13] M. Scarpiniti, D. Comminiello, G. Scarano, R. Parisi, and A. Uncini, "Steady-State Performance of Spline Adaptive Filters," *IEEE Trans. Signal Process.*, vol. 64, no. 4, pp. 816–828, 2016.

[14] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proc. of the 5th international conference on Information processing in sensor networks (IPSN'06)*. ACM, 2006, pp. 168–176.

[15] X. Zhao and A. H. Sayed, "Asynchronous Adaptation and Learning Over Networks Part I: Modeling and Stability Analysis," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 811–826, Feb 2015.

[16] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini, "Hammerstein uniform cubic spline adaptive filters: Learning and convergence properties," *Signal Processing*, vol. 100, pp. 112 – 123, 2014.