A DIFFUSION KERNEL LMS ALGORITHM FOR NONLINEAR ADAPTIVE NETWORKS

Symeon Chouvardas, Moez Draief

Mathematical and Algorithmic Sciences Lab,
Huawei France R&D,
Paris, France.
symeon.chouvardas,moez.draief@huawei.com

ABSTRACT

This work presents a distributed algorithm for nonlinear adaptive learning. In particular, a set of nodes obtain measurements, sequentially one per time step, which are related via a nonlinear function; their goal is to collectively minimize a cost function by employing a diffusion based Kernel Least Mean Squares (KLMS). The algorithm follows the Adapt Then Combine mode of cooperation. Moreover, the theoretical properties of the algorithm are studied and it is proved that under certain assumptions the algorithm suffers a no regret bound. Finally, comparative experiments verify that the proposed scheme outperforms other variants of the LMS.

***Index Terms*—** Adaptive Networks, Diffusion, RKHS, Kernel LMS.

1. INTRODUCTION

In recent years, the interest in the topic of *distributed learning* and inference has grown rapidly. This is mainly due to the constantly increasing requirements for memory and computational resources, demanded by modern applications, so as to cope with the huge amount of available data. These data "spring" from several sources/applications, such as communication, imaging, medical platforms as well as social-networking sites, e.g., [1]. A natural way, to deal with the large number of data, which need to be processed, is to split the problem into subproblems and resort to distributed operations [2, 3]. Thus, the development of algorithms dealing with such scenarios, where the data are not available in a single location but are instead spread out over multiple locations, becomes essential.

An important application within the distributed learning context is the one of *distributed adaptive learning*, [4]. In a nutshell, this problem considers a decentralized network consisting of nodes interested in performing a specific task, which can be, for instance, parameter estimation, classification, etc. The nodes constantly obtain new measurements and they continuously adapt and learn; this gives them the capability to track and adapt to changes in the environment. On top of that, it is assumed that there is no central node which could perform all the necessary operations and, so, the nodes act as independent learners and perform the computations by themselves. Finally, the task of interest is considered to be common or similar across the nodes and, to that direction, they cooperate with each other. Cooperation has been proved to be beneficial to the learning process since it improves the learning performance, [4].

This paper is concerned with the problem of distributed adaptive learning in Reproducing Kernel Hilbert spaces (RHKS). To be more specific, we consider an ad–hoc network the nodes of which obtain input/output measurements, sequentially, one per time step, related via a *nonlinear* unknown function. To cope with this nonlinearity we resort to the family of the kernel–based algorithms for nonlinear adaptive filtering. In particular, the proposed algorithm belongs to the Kernel LMS (KLMS) algorithmic family and follows the diffusion rationale for cooperation among the nodes.

Related Work: Several studies for distributed adaptive estimation of linear systems have been proposed in the literature. These include diffusion based algorithms, e.g., [5, 6, 7], consensus ones, e.g., [8, 9], as well as algorithms for multitask learning [10, 11]. The problem of non–linear adaptive estimation in RKHS has been studied, e.g., [12, 13, 14, 15]. A recent study, which considers the problem of nonlinear adaptive filtering in distributed networks, can be found in [16]. The major differences of this paper with our work are summarized in the sequel. First, the authors consider a predefined dictionary, which essentially makes the dimension of the problem finite and equal to the number of elements of the dictionary. On the contrary, here, we consider the general case, where the dictionary is allowed to grow as time increases, and we present a more general form of the algorithm. Furthermore, here, we study for the first time the theoretical properties of the Diffusion Kernel LMS (DKLMS) and we derive regret bounds for the proposed scheme.

Contributions: In this paper, we propose a novel nonlinear distributed algorithm for adaptive learning. In particular, we propose a KLMS algorithm, which follows the diffusion rationale. The Adapt Then Combine mode of cooperation among the nodes is followed. To be more specific, we assume that the nodes obtain measurements, which arrive sequentially

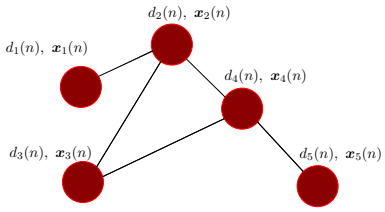

Fig. 1. An ad–hoc Network

and are related via a nonlinear system. The goal is the minimization of the expected value of the networkwise discrepancy between the desired output and the estimated one. To that direction, at each step, the nodes: a) perform a local update step exploiting their most recent measurements, b) cooperate with each other, in order to enhance their estimates. Comparative experiments illustrate that the proposed scheme outperforms other LMS variants and the theoretical properties of the proposed scheme are discussed.

Notation: Lowercase and uppercase boldfaced letters stand for vectors and matrices respectively. The symbol $\mathbb{R}$ stands for the set of real numbers and $\mathbb{N}$ for the set of nonnegative integers. $\mathcal{H}$ denotes an infinite dimensional Hilbert space equipped with an inner product denoted by $\langle f_1, f_2\rangle,\ \forall f_1,\ f_2\ \in\ \mathcal{H}$; the induced norm is given by $\|f\| = \sqrt{\langle f, f\rangle}$. Given a set $\mathcal{S}$, with the term $|\mathcal{S}|$ we denote its cardinality.

2. PROBLEM STATEMENT

We consider an ad–hoc network illustrated in Fig. 1, consisting of K nodes. Each node, $k \in \mathcal{N} := \{1, \ldots, K\}$, at each discrete time instance $n \in \mathbb{N}$, has access to a scalar $d_k(n) \in \mathbb{R}$ and a vector $\boldsymbol{x}_k(n) \in \mathbb{R}^m$, which are related via

$$d_k(n) = f^o(\boldsymbol{x}_k(n)) + v_k(n), \tag{1}$$

where $f^o : \mathbb{R}^m \mapsto \mathbb{R}$ is an unknown yet common to all the nodes function belonging to the Hilbert space $\mathcal{H}$ and the term $v_k(n)$ stands for the additive noise process. The overall goal is the estimation of a function, f, which minimizes the cost:

$$J(f) = \sum_{k\in\mathcal{N}} \mathbb{E}\{(d_k(n) - f(\boldsymbol{x}_k(n)))^2\}, \tag{2}$$

in a *distributed* and *collaborative* fashion; that is the nodes want to minimize the cost (2) by relying solely on local processing as well as interactions with their neighbors.

2.1. Linear Diffusion LMS

In order to help the reader grasp the concept of the diffusion LMS, in this section we describe the linear scenario, i.e., the one where the function to be estimated is a vector, say $\boldsymbol{w}_* \in \mathbb{R}^m$, and (1) essentially becomes:

$$d_k(n) = \boldsymbol{w}_*^T \boldsymbol{x}_k(n) + v_k(n). \tag{3}$$

The cost function to be minimized in that case can be written as follows:

$$J(\boldsymbol{w}) = \sum_{k\in\mathcal{N}} \mathbb{E}\{\big(d_k(n) - \boldsymbol{w}^T \boldsymbol{x}_k(n)\big)^2\}. \tag{4}$$

The cost (4) includes information coming from the whole network and in order to minimize it, global knowledge is required. Nevertheless, in distributed and decentralized learning each node can only interact and exchange information with its neighborhood which will be denoted by $\mathcal{N}_k,\ \forall k \in \mathcal{N}$. A fully distributed algorithm, which can be employed for the estimation of $\boldsymbol{w}_*$ is the diffusion LMS (see for example [4, 7]). The starting point of this scheme is a modification of the steepest–descent method, which is properly reformulated so as to enable distributed operations and to avoid any global computation (the interested reader is referred to [4]). In addition, the instantaneous approximation is adopted, according to which the statistical values are substituted by their instantaneous ones, e.g., [17]. Each node $k \in \mathcal{N}$ updates the estimate $\boldsymbol{w}_k(n)$ at each time step according to the following iterative scheme:

$$\boldsymbol{w}'_k(n) = \boldsymbol{w}_k(n-1) + \mu_k e_k(n) \boldsymbol{x}_k(n) \tag{5}$$

$$\boldsymbol{w}_k(n) = \sum_{l\in\mathcal{N}_k} a_{k,l} \boldsymbol{w}'_l(n), \tag{6}$$

where $e_k(n) := d_k(n) - \boldsymbol{w}_k^T(n-1)\boldsymbol{x}_k(n)$ and μ_k is the step size. Furthermore, $a_{k,l}$ stand for combination coefficients, which have the following properties $a_{k,l} = 0,\ l \notin \mathcal{N}_k,\ a_{k,l} > 0,\ l \in \mathcal{N}_k, \sum_{l\in\mathcal{N}} a_{k,l} = 1$. A common choice, among others, for choosing these coefficients is the Metropolis rule, in which the weights equal to:

$$a_{k,l} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_k|, |\mathcal{N}_l|\}}, & \text{if } l \in \mathcal{N}_k, \text{ and } l \neq k \\ 1 - \sum_{l\in\mathcal{N}_k \setminus k} a_{k,l}, & \text{if } l = k \\ 0, & \text{otherwise.} \end{cases}$$

The intuition behind the scheme presented in (5), (6), can be summarized as follows. In the first step, node k updates its estimate using an LMS based update (adaptation step) exploiting local information. In the sequel, k cooperates with its neighborhood by combining their intermediate estimates to obtain its updated estimate $\boldsymbol{w}_k(n)$. The weights $a_{k,l}$ assign a non–negative weight to the estimates received by the neighborhood, whereas they are equal to zero for the rest of the nodes. Hence, each node *aggregates* the information received by the neighborhood. This scheme is also known as Adapt Then Combine (ATC) diffusion strategy.

2.2. Centralized Kernel LMS

2.2.1. Preliminaries

Now, let us provide with a few elementary properties of the RKHS, which will be used in the sequel. Throughout this section the node subscript will be suppressed since we will

describe properties of centralized learning. We consider a real Hilbert space $\mathcal{H}$ comprising functions defined on $\mathbb{R}^m$; that is $f : \mathbb{R}^m \mapsto \mathbb{R}$. The function $\kappa : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$ will be called a reproducing kernel of $\mathcal{H}$ if the following properties hold:

- $\forall \boldsymbol{x} \in \mathbb{R}^m$ the function $\kappa(\boldsymbol{x}, \cdot)$ belongs to $\mathcal{H}$.
- $\forall \boldsymbol{x} \in \mathbb{R}^m,\ f \in \mathcal{H}$, it holds that $f(\boldsymbol{x}) = \langle f, \kappa(\boldsymbol{x}, \cdot)\rangle$.

If these properties hold then $\mathcal{H}$ is called a Reproducing Kernel Hilbert Space [18, 19]. A typical example is the Gaussian kernel with definition: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\beta\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2),\ \beta > 0$. A very important property, which will be exploited in the sequel states that points in the RKHS can be written as follows:

$$f = \sum_{n=0}^{\infty} \alpha_n \kappa(\boldsymbol{x}(n), \cdot), \tag{7}$$

where $\alpha_n \subset \mathbb{R}$. Finally, the reproducing kernel is continuous, symmetric and positive-definite.

2.2.2. Kernel LMS

Kernel LMS, which was originally proposed in [20], is a generalization of the original LMS algorithm, which utilizes the transformed input, i.e., $\kappa(\boldsymbol{x}(n), \cdot)$, at each iteration step. Put in mathematical terms, the recursion of the KLMS is given by:

$$e(n) = d(n) - \langle f_{n-1}, \kappa(\boldsymbol{x}(n), \cdot)\rangle \tag{8}$$

$$f_n = f_{n-1} + \mu e(n)\kappa(\boldsymbol{x}(n), \cdot). \tag{9}$$

Since the space $\mathcal{H}$ may be infinite dimensional, it may be difficult to have direct access to the transformed input data and the function f_n. However, if we go back to (1) and forget the distributed aspect for now, we can see that the quantity of interest is $f(\boldsymbol{x}(n))$, which can be computed exploiting (7). In particular, following similar steps as in [20] it can be shown that the KLMS recursion can be equivalently written:

$$e(n) = d(n) - \mu \sum_{i=1}^{n-1} e(i)\kappa(\boldsymbol{x}(n), \boldsymbol{x}(i)) \tag{10}$$

$$f_n(\boldsymbol{x}(n)) = \mu \sum_{i=1}^{n} e(i)\kappa(\boldsymbol{x}(n), \boldsymbol{x}(i)). \tag{11}$$

Note that this reformulation is very convenient as it computes the response of the estimated function to the input, without any need to estimate the function itself.

3. THE DIFFUSION KERNEL LMS

In this section we describe the proposed algorithm together with its theoretical properties. Recall the problem under consideration, discussed in Section 2. Our goal, here, is to bring together the tools described in Sections 2.1, 2.2 and derive a Kernel based LMS algorithm suitable for distributed operation. Our starting point will be the ATC–LMS described in (5)-(6) and we will employ the non–linear transformation on the input (similarly to (8), (9)). The resulting recursion $\forall k \in \mathcal{N}$ is:

$$f'_{k,n} = f_{k,n-1} + \mu_k e_k(n)\kappa(\boldsymbol{x}_k(n), \cdot) \tag{12}$$

$$f_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} f'_{l,n}, \tag{13}$$

where $e_k(n)$ is defined similarly to (8). Despite the fact that this seems a trivial generalization of (5), (6), as we have already discussed previously, one cannot resort directly to this form of iterations, since access to the transformed data may not be possible.

Exploiting the lemma, which will be presented shortly, we can bypass the aforementioned problem, by deriving the inner product between the obtained function and the transformed input vector in a closed form. Before we proceed, let us introduce some notation. The networkwise function at time n is denoted by $\underline{f}_n := [f_{1,n}, \ldots, f_{K,n}]^T \in \boldsymbol{\mathcal{H}}$, where the Cartesian product $\boldsymbol{\mathcal{H}} := \underbrace{\mathcal{H} \times \ldots \times \mathcal{H}}_{K \text{ times}}$. Similarly, we define the networkwise input: $\kappa(\underline{\boldsymbol{x}}(n), \cdot) = [\kappa(\boldsymbol{x}_1(n), \cdot), \ldots, \kappa(\boldsymbol{x}_K(n), \cdot)]^T \in \boldsymbol{\mathcal{H}}$ and $\boldsymbol{g}(n) = [\mu_1 e_1(n), \ldots, \mu_K e_K(n)]^T \in \mathbb{R}^m$. Finally, we gather the combination coefficients to the $K \times K$ matrix $\boldsymbol{A}$, the k, l–th entry of which contains $a_{k,l}$. It can be readily shown that (12)-(13) can be written for the whole network in the following compact form:

$$\underline{f}_n = \boldsymbol{A}\left(\underline{f}_{n-1} + \boldsymbol{g}(n)\kappa(\underline{\boldsymbol{x}}(n), \cdot)\right). \tag{14}$$

Lemma 1 *Assume that $f_{k,0} = 0,\ \forall k \in \mathcal{N}$. Then equation* (14) *can be equivalently written:*

$$\underline{f}_n = \sum_{i=1}^{n} \boldsymbol{A}^{n-i+1} \boldsymbol{g}(i)\kappa(\underline{\boldsymbol{x}}(i), \cdot). \tag{15}$$

Hence, the vector of responses, $\tilde{\boldsymbol{d}}(n) := [\tilde{d}_1(n), \ldots, \tilde{d}_K(n)]^T$, at time instance n, is given by $\tilde{\boldsymbol{d}}(n) = \underline{f}_n(\kappa(\underline{\boldsymbol{x}}(n), \cdot)) = \sum_{i=1}^{n} \boldsymbol{A}^{n-i+1} \boldsymbol{g}(i)\kappa(\underline{\boldsymbol{x}}(i), \underline{\boldsymbol{x}}(n))$ ■

The proof, which follows mathematical induction, is omitted due to lack of space and will be presented elsewhere.

Remark 1 ***Coefficient Reduction over Time:*** *If we take a closer look on* (15) *it can be seen that the number of coefficients one has to store as well as the required number of operations grow as time evolves. Several sophisticated techniques, which set most of the coefficients to zero while avoiding performance degradation, have been proposed in the literature, e.g., [21]. As it will become apparent in the simulations section, here we adopt a simple method; that is, we apply a buffer of size L. In that case, we store the L most recent coefficients and* (15) *becomes $\underline{f}_n = \sum_{i=\max(1,n-L+1)}^{n} \boldsymbol{A}^{n-i+1} \boldsymbol{g}(i)\kappa(\underline{\boldsymbol{x}}(i), \cdot)$.*

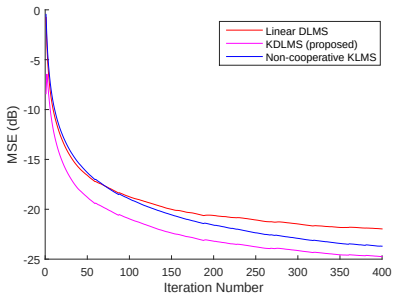

Fig. 2. Average MSE for the first experiment

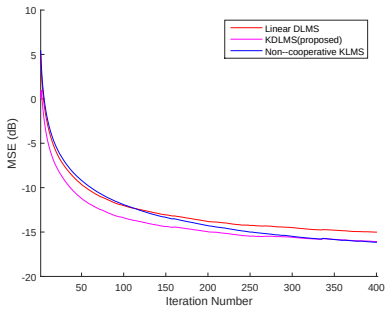

Fig. 3. Average MSE for the second experiment

Remark 2 ***Coefficient Reduction over Space:*** *It can be shown (see [4, Appendix E]) that the k, l–th entry of the i–th power of the matrix $\boldsymbol{A}$ equals to: $[\boldsymbol{A}^i]_{kl} = \sum_{j_1}^{K}\sum_{j_2}^{K}\dots\sum_{j_{i-1}}^{K} a_{kj_1}a_{kj_2}\dots a_{j_{i-1}l}$. From the last relation it is not difficult to obtain that the node k exploits information from nodes which do not belong to its neighorhood. However, this does not break the rules of decentralized learning, since it also holds that $[\boldsymbol{A}^i]_{kl}$ will be nonzero iff the distance, measured in hops, between k, l is smaller or equal than i hops. Hence, the nodes can send their input vectors to their neighbors, which in turn will forward them to their neighbors and so on. This increases the network load and one can avoid it by setting some of the weights to zero, as discussed on Remark 1. A simple strategy is to set to zero all the coefficients that belong to nodes which don't belong to the neighborhood.*

3.1. Theoretical Properties

In the sequel, we will present the regret bound of the proposed scheme and in particular we will show that this grows sublinearly with the time.

Theorem 1 *Under certain assumptions about the boundness of the input, the step-size and the combination weights, the networkwise regret is bounded by*

$$\sum_{i=1}^{N}\sum_{k\in\mathcal{N}}\left(J_{k,i}(f_{k,i-1}) - J_{k,i}(g)\right) \leq \gamma\sqrt{N} + \delta, \forall g \in \mathcal{H} \quad (16)$$

where $J_{k,n}(f) = \frac{1}{2}(d_k(n) - f(\boldsymbol{x}_k(n)))^2$ and γ, δ are positive constants. ■

Proof: The proof is omitted due to lack of space and will be presented elsewhere.

4. SIMULATIONS

In this section, the performance of the proposed algorithm is validated within the distributed nonlinear adaptive filtering framework. We consider a network comprising $K = 10$ nodes and a distributed version of the problem studied in [22, 23], for which the input and the output are related via:

$$y_k(n) = \frac{y_k(n-1)}{1 + y_k^2(n-1)} + x_k^3(n),\ d_k(n) = y_k(n) + v_k(n),$$

where $v_k(n)$ is Gaussian with variance 10^{-3} and the input $x_k(n)$ is also Gaussian with variance $0.1\chi_k$, where $\chi_k \in [0.5, 1], \forall k \in \mathcal{N}$ with respect to the Uniform distribution. We compare the proposed algorithm with: a) the linear diffusion LMS, b) the non–cooperative KLMS , i.e., the KLMS where the nodes do not cooperate with each other. For the Kernel based algorithms we employ the Gaussian Kernel with $\alpha = 1.1$ and we choose a step–size equal to $\mu = 0.6$ for all the algorithms. Furthermore, the combination weights are chosen with respect to the Metropolis rule, the buffer size L at each node equals to 100 and we only take into consideration information that is coming from the single hop neighbors. Finally, the adopted performance metric is the average MSE, with definition $MSE(n) = 1/K\sum_{k\in\mathcal{N}}(d_k(n) - f_{k,n}(\boldsymbol{x}_k(n))$. As it can be seen from Fig. 2 the KDLMS outperforms the other LMS variants, since it converges faster to a lower error floor compared to them. In the second experiment the setup is similar to the previous one albeit here we increase the variance of the noise, which now equals to 10^{-1}. Fig. 3 illustrates that the enhanced performance of the KDLMS, compared to the other algorithms, is retained in this scenario as well.

5. CONCLUSIONS AND FUTURE RESEARCH

In this paper, a novel Kernel based Diffusion LMS, suitable for non–linear distributed adaptive filtering was proposed. The theoretical properties of the algorithm were discussed and the performance of the scheme was tested against other adaptive strategies. Future research focuses on accelerating the convergence speed by utilizing more data per iteration, as well as investigating sophisticated strategies, which reduce the number of coefficients by storing the most "informative" ones.

6. REFERENCES

[1] Konstantinos Slavakis, Georgios Giannakis, and Gonzalo Mateos, "Modeling and optimization for big data analytics:(statistical) learning tools for our era of data deluge," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 18–31, 2014.

[2] Cheng Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun, "Map-reduce for machine learning on multicore," *Advances in neural information processing systems*, vol. 19, pp. 281, 2007.

[3] Paul Zikopoulos, Chris Eaton, et al., *Understanding big data: Analytics for enterprise class hadoop and streaming data*, McGraw-Hill Osborne Media, 2011.

[4] Ali H Sayed, "Diffusion adaptation over networks," *Academic Press Library in Signal Processing*, vol. 3, pp. 323–454, 2013.

[5] Symeon Chouvardas, Konstantinos Slavakis, and Sergios Theodoridis, "Adaptive robust distributed learning in diffusion sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4692–4707, 2011.

[6] Cassio G Lopes and Ali H Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

[7] Federico S Cattivelli and Ali H Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, 2010.

[8] Ioannis D Schizas, Gonzalo Mateos, and Georgios B Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365–2382, 2009.

[9] Gonzalo Mateos, Ioannis D Schizas, and Georgios B Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *Signal Processing, IEEE Transactions on*, vol. 57, no. 11, pp. 4583–4588, 2009.

[10] Jie Chen, Cédric Richard, and Ali H Sayed, "Multitask diffusion adaptation over networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4129–4144, 2014.

[11] Jorge Plata-Chaves, Nikola Bogdanovic, and Kostas Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3448–3460, 2015.

[12] Pantelis Bouboulis and Sergios Theodoridis, "Extension of wirtinger's calculus to reproducing kernel hilbert spaces and the complex kernel LMS," *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 964–978, 2011.

[13] Pantelis Bouboulis, Sergios Theodoridis, and Michael Mavroforakis, "The augmented complex kernel LMS," *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4962–4967, 2012.

[14] Konstantinos Slavakis, Sergios Theodoridis, and Isao Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 2781–2796, 2008.

[15] Konstantinos Slavakis, Sergios Theodoridis, and Isao Yamada, "Adaptive constrained learning in reproducing kernel hilbert spaces: the robust beamforming case," *IEEE Transactions on Signal Processing*, vol. 57, no. 12, pp. 4744–4764, 2009.

[16] Wei Gao, Jie Chen, Cédric Richard, and Jianguo Huang, "Diffusion adaptation over networks with kernel least-mean-square," *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE International Workshop on, 2015 (submitted).*

[17] Simon S Haykin, *Adaptive filter theory*, Pearson Education India, 2008.

[18] Alex J Smola and Bernhard Schölkopf, *Learning with kernels*, Citeseer, 1998.

[19] Sergios Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press, 2015.

[20] Weifeng Liu, Puskal P Pokharel, and Jose C Principe, "The kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, 2008.

[21] Wei Gao, Jie Chen, Cedric Richard, and Jianguo Huang, "Online dictionary learning for kernel LMS," *IEEE Transactions on Signal Processing*, vol. 62, no. 11, pp. 2765–2777, 2014.

[22] Kumpati S Narendra and Kannan Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[23] Danilo P Mandic, "A generalized normalized gradient descent algorithm," *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 115–118, 2004.