Online Distributed Learning Over Networks in RKH Spaces Using Random Fourier Features

Pantelis Bouboulis, *Member, IEEE,* Symeon Chouvardas, *Member, IEEE,* and Sergios Theodoridis, *Fellow, IEEE*

***Abstract*—We present a novel diffusion scheme for online kernel-based learning over networks. So far, a major drawback of any online learning algorithm, operating in a reproducing kernel Hilbert space (RKHS), is the need for updating a growing number of parameters as time iterations evolve. Besides complexity, this leads to an increased need of communication resources, in a distributed setting. In contrast, we propose to approximate the solution as a fixed-size vector (of larger dimension than the input space) using the previously introduced framework of Random Fourier Features. This paves the way to use standard linear combine-then-adapt techniques. To the best of our knowledge, this is the first time that a complete protocol for distributed online learning in RKHS is presented. Conditions for asymptotic convergence and boundness of the networkwise regret are also provided. The simulated tests illustrate the performance of the proposed scheme.**

***Index Terms*—Diffusion, KLMS, Distributed, RKHS, online learning.**

I. Introduction

THE topic of distributed learning, has grown rapidly over the last years. This is mainly due to the exponentially increasing volume of data, that leads, in turn, to increased requirements for memory and computational resources. Typical applications include sensor networks, social networks, imaging, databases, medical platforms, e.t.c., [1]. In most of those, the data cannot be processed on a single processing unit (due to memory and/or computational power constraints) and the respective learning/inference problem has to be split into subproblems. Hence, one has to resort to distributed algorithms, which operate on data that are not available on a single location but are instead spread out over multiple locations, e.g., [2], [3], [4].

In this paper, we focus on the topic of *distributed online learning* and in particular to non linear parameter estimation and classification tasks. More specifically, we consider a decentralized network which comprises nodes, that observe data generated by a non linear model in a sequential fashion. Each node communicates its own estimates of the unknown parameters to its neighbors and exploits simultaneously a) the information that it receives and b) the observed datum, at each time instant, in order to update the associated estimates. Furthermore, no assumptions are made regarding the presence of a central node, which could perform all the necessary operations. Thus, the nodes act as independent learners and perform the computations by themselves. Finally, the task of interest is considered to be common across the nodes and, thus, cooperation among each other is meaningful and beneficial, [5], [6].

The problem of linear online estimation has been considered in several works. These include diffusion-based algorithms, e.g., [7], [8], [9], ADMM-based schemes, e.g., [10], [11], as well as consensus-based ones, e.g., [12], [13]. The multitask learning problem, in which there are two or more parameter vectors to be estimated, has also been treated, e.g., [14], [15]. The literature on online distributed classification is more limited; in [16], a batch distributed SVM algorithm is presented, whereas in [17], a diffusion based scheme suitable for classification is proposed. In the latter, the authors study the problem of distributed online learning focusing on strongly-convex risk functions, such as the logistic regression loss, which is suitable to tackle classification tasks. The nodes of the network cooperate via the diffusion rationale. In contrast to the vast majority of works on the topic of distributed online learning, which assume a linear relationship between input and output measurements, in this paper we tackle the more general problem, i.e., the distributed online *non–linear* learning task. To be more specific, we assume that the data are generated by a model $y = f(\boldsymbol{x})$, where f is a non-linear function that lies in a *Reproducing Kernel Hilbert Space* (RKHS). These are inner-product function spaces, generated by a specific kernel function, that have become popular models for non-linear tasks, since the introduction of the celebrated Support Vectors Machines (SVM) [18], [19], [20], [6].

Although there have been methods that attempt to generalize linear online distributed strategies to the non-linear domain using RKHS, mainly in the context of the Kernel Least Mean Squares (KLMS) e.g., [21], [22], [23], these have major drawbacks. In [21] and [23], the estimation of f, at each node, is given as an increasingly growing sum of kernel functions centered at the observed data. Thus, a) each node has to transmit the entire sum at each time instant to its neighbors and b) the node has to fuse together all sums received by its neighbors to compute the new estimation. Hence, both the communications load of the entire network as well as the computational burden at each node grow linearly with time. Clearly, this is impractical for real life applications. In contrast, the method of [22] assumes that these growing sums are limited by a sparsification strategy; how this can be achieved is left for the future. Moreover, the aforementioned methods offer no theoretical results regarding the consensus of the network. In this work, we present a complete protocol for

P. Bouboulis and S. Theodoridis are with the Department of Informatics and Telecommunications, University of Athens, Greece, e-mails: panbouboulis@gmail.com, stheodor@di.uoa.gr.

S. Chouvardas is with the Mathematical and Algorithmic Sciences Lab France Research Center, Huawei Technologies Co., Ltd., e-mail: symeon.chouvardas@huawei.com

distributed online non-linear learning for both regression and classification tasks, overcoming the aforementioned problems. Moreover, we present theoretical results regarding network-wise consensus and regret bounds. The proposed framework offers fixed-size communication and computational load as time evolves. This is achieved through an efficient approximation of the growing sum using the random Fourier features rationale [24]. To the best of our knowledge, this is the first time that such a method appears in the literature.

Section II presents a brief background on kernel online methods and summarizes the main tools and notions used in this manuscript. The main contributions of the paper are presented in section III. The proposed method, the related theoretical results and extensive experiments can be found there. Section IV presents a special case of the proposed framework for the case of a single node. In this case, we demonstrate how the proposed scheme can be seen as a fixed-budget alternative for online kernel based learning (solving the problem of the growing sum). Finally, section V offers some concluding remarks. In the rest of the paper, boldface symbols denote vectors, while capital letters are reserved for matrices. The symbol $\otimes$ denotes the Kronecker product of matrices and the symbol $\cdot^T$ the transpose of the respective matrix or vector. Finally, the symbol $\|\cdot\|$ refers to the respective ℓ_2 matrix or vector norm.

II. Preliminaries

A. RKHS

Reproducing Kernel Hilbert Spaces (RKHS) are inner product spaces of functions defined on X, whose respective point evaluation functional, i.e., $T_x : \mathcal{H} \to X : T_x(f) = f(x)$, is linear and continuous for every $x \in X$. This is usually portrayed by the *reproducing property* [18], [6], [25], which links inner products in $\mathcal{H}$ with a specific (semi-)positive definite kernel function κ defined on $X \times X$ (associated with the space $\mathcal{H}$). As $\kappa(\cdot, x)$ lies in $\mathcal{H}$ for all $x \in X$, the reproducing property declares that $\langle \kappa(\cdot, y), \kappa(\cdot, x)\rangle_{\mathcal{H}} = \kappa(x, y)$, for all $x, y \in X$. Hence, linear tasks, defined on the high dimensional space, $\mathcal{H}$, (whose dimensionality can also be infinite) can be equivalently viewed as non-linear ones on the, usually, much lower dimensional space, X, and vice versa. This is the essence of the so called *kernel trick*: Any kernel-based learning method can be seen as a two step procedure, where firstly the original data are transformed from X to $\mathcal{H}$, via an implicit map, $\Phi(x) = \kappa(\cdot, x)$, and then linear algorithms are applied to the transformed data. There exist a plethora of different kernels to choose from in the respective literature. In this paper, we mostly focus on the popular Gaussian kernel, i.e., $\kappa(\boldsymbol{x}, \boldsymbol{y}) = e^{\|\boldsymbol{x}-\boldsymbol{y}\|^2/(2\sigma^2)}$, for some predefined $\sigma \in \mathbb{R}$, although any other shift invariant kernel can be adopted too.

Another important feature of RKHS is that any regularized ridge regression task, defined on $\mathcal{H}$, has a unique solution, which can be written in terms of a finite expansion of kernel functions centered at the training points. Specifically, given the set of training points $\{(x_n, y_n), n = 1, \dots, N, x_n \in X, y_n \in \mathbb{R}\}$, the *representer theorem* [26], [18], states that the unique minimizer, $f_* \in \mathcal{H}$, of $\sum_{n=1}^N l(f(x_n), y_n) + \lambda\|f\|^2_{\mathcal{H}}$, admits a representation of the form $f_* = \sum_{n=1}^N a_n \kappa(\cdot, x_n)$, where l is any convex loss function that measures the error between the actual system's outputs, y_n, and the estimated ones, $f(x_n)$, and $\|\cdot\|_{\mathcal{H}}$ is the norm induced by the inner product.

B. Kernel Online Learning

The aforementioned properties have rendered RKHS a popular tool for addressing non linear tasks both in batch and online settings. Besides the widely adopted application on SVMs, in recent years there has been an increased interest on non linear online tasks around the squared error loss function. Hence, there have been kernel-based implementations of LMS [27], [28], RLS [29], [30], APSM [31], [32] and other related methods [33], as well as online implementations of SVMs [34], focusing on the primal formulation of the task. Henceforth, we will consider online learning tasks based on the training sequences of the form $\mathcal{D} = \{(\boldsymbol{x}_n, y_n),\ n = 1, 2, \dots\}$, where $\boldsymbol{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$. The goal of the assumed learning tasks is to learn a non-linear input-output dependence, $y = f(\boldsymbol{x})$, $f \in \mathcal{H}$, so that to minimize a preselected cost. Note that these types of tasks include both classification (where $y_n = \pm 1$) and regression problems (where $y_n \in \mathbb{R}$). Moreover, in the online setting, the data are assumed to arrive sequentially.

As a typical example of these tasks, we consider the KLMS, which is one of the simplest and most representative methods of this kind. Its goal is to learn f, so that to minimize the MSE, i.e., $\mathcal{L}(f) = E[(y - f(\boldsymbol{x}))^2]$. Computing the gradient of $\mathcal{L}$ and estimating it via the current set of observations (in line with the stochastic approximation rationale, e.g., [6]), the estimate at the next iteration, employing the gradient descent method, becomes $f_n = f_{n-1} + \mu\varepsilon_n\kappa(\boldsymbol{x}_n, \cdot)$, where $\varepsilon_n = y_n - f_{n-1}(\boldsymbol{x}_n)$ and μ is the step-size (see, e.g., [6], [35], [36] for more). Assuming that the initial estimate is zero, the solution after $n-1$ steps turns out to be

$$f_{n-1} = \sum_{i=1}^{n-1} \alpha_i \kappa(\cdot, \boldsymbol{x}_i), \tag{1}$$

where $\alpha_i = \mu\varepsilon_i$. Observe that this is in line with the representer theorem. Similarly, the system's output can be estimated as $f_{n-1}(\boldsymbol{x}_n) = \sum_{i=1}^{n-1} \alpha_i \kappa(\boldsymbol{x}_n, \boldsymbol{x}_i)$. Clearly, this linear expansion grows indefinitely as n increases; hence the original form of KLMS is impractical. Typically, a sparsification strategy is adopted to bound the size of the expansion [37], [38], [39]. In these methods, a specific criterion is employed to decide whether a particular point, $\boldsymbol{x}_n$, is to be included to the expansion, or (if that point is discarded) how its respective output y_n can be exploited to update the remaining weights of the expansion. There are also methods that can remove specific points from the expansion, if their information becomes obsolete, in order to increase the tracking ability of the algorithm [40].

C. Approximating the Kernel with random Fourier Features

Usually, kernel-based learning methods involve a large number of kernel evaluations between training samples. In the batch mode of operation, for example, this means that a large kernel matrix has to be computed, increasing the computational cost of the method significantly. Hence, to alleviate the computational burden, one common approach is to use some sort of approximation of the kernel evaluation. The most popular techniques of this category are the Nyström method [41], [42] and the random Fourier features approach [24], [43]; the latter fits naturally to the online setting. Instead of relying on the implicit lifting, Φ, provided by the kernel trick, Rahimi and Recht in [24] proposed to map the input data to a finite-dimensional Euclidean space (with dimension lower than $\mathcal{H}$ but larger than the input space) using a randomized feature map $\boldsymbol{z}_\Omega : \mathbb{R}^d \to \mathbb{R}^D$, so that the kernel evaluations can be approximated as $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) \approx \boldsymbol{z}_\Omega(\boldsymbol{x}_n)^T \boldsymbol{z}_\Omega(\boldsymbol{x}_m)$. The following theorem plays a key role in this procedure.

Theorem 1. *Consider a shift-invariant positive definite kernel $\kappa(\boldsymbol{x}-\boldsymbol{y})$ defined on $\mathbb{R}^d$ and its Fourier transform $p(\boldsymbol{\omega}) = \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d}\kappa(\boldsymbol{\delta})e^{-i\boldsymbol{\omega}^T\boldsymbol{\delta}}d\boldsymbol{\delta}$, which (according to Bochner's theorem) it can be regarded as a* ***probability density*** *function. Then, defining $z_{\boldsymbol{\omega},b}(\boldsymbol{x}) = \sqrt{2}\cos(\boldsymbol{\omega}^T\boldsymbol{x}+b)$, it turns out that*

$$\kappa(\boldsymbol{x}-\boldsymbol{y}) = E_{\boldsymbol{\omega},b}[z_{\boldsymbol{\omega},b}(\boldsymbol{x})z_{\boldsymbol{\omega},b}(\boldsymbol{y})], \tag{2}$$

where $\boldsymbol{\omega}$ is drawn from p and b from the uniform distribution on $[0, 2\pi]$.

Following Theorem 1, we choose to approximate $\kappa(\boldsymbol{x}_n - \boldsymbol{x}_m)$ using D random Fourier features, $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \dots, \boldsymbol{\omega}_D$, (drawn from p) and D random numbers, $b_1, b_2, \dots, b_D$ (drawn uniformly from $[0, 2\pi]$) that define a sample average:

$$\kappa(\boldsymbol{x}_n - \boldsymbol{x}_m) \approx \frac{1}{D}\sum_{i=1}^{D} z_{\boldsymbol{\omega}_i,b_i}(\boldsymbol{x}_m) z_{\boldsymbol{\omega}_i,b_i}(\boldsymbol{x}_n). \tag{3}$$

Evidently, the larger D is, the better this approximation becomes (up to a certain point). Details on the quality of this approximation can be found in [24], [43], [44], [45]. We note that for the Gaussian kernel, which is employed throughout the paper, the respective Fourier transform is

$$p(\boldsymbol{\omega}) = \left(\sigma/\sqrt{2\pi}\right)^D e^{-\frac{\sigma^2\|\boldsymbol{\omega}\|^2}{2}}, \tag{4}$$

which is actually the multivariate Gaussian distribution with mean $\boldsymbol{0}_D$ and covariance matrix $\frac{1}{\sigma^2}\boldsymbol{I}_D$.

We will demonstrate how this method can be applied using the KLMS paradigm. To this end, we define the map $\boldsymbol{z}_\Omega : \mathbb{R}^d \to \mathbb{R}^D$ as follows:

$$\boldsymbol{z}_\Omega(\boldsymbol{u}) = \sqrt{\frac{2}{D}}\begin{pmatrix}\cos(\boldsymbol{\omega}_1^T\boldsymbol{u}+b_1)\\ \vdots \\ \cos(\boldsymbol{\omega}_D^T\boldsymbol{u}+b_D)\end{pmatrix}, \tag{5}$$

where Ω is the $(d+1)\times D$ matrix defining the random Fourier features of the respective kernel, i.e.,

$$\Omega = \begin{pmatrix}\boldsymbol{\omega}_1 & \boldsymbol{\omega}_2 & \dots & \boldsymbol{\omega}_D\\ b_1 & b_2 & \dots & b_D\end{pmatrix}, \tag{6}$$

provided that $\boldsymbol{\omega}$'s and b's are drawn as described in theorem 1. Employing this notation, it is straightforward to see that (3) can be recast as $\kappa(\boldsymbol{x}_n - \boldsymbol{x}_m) \approx \boldsymbol{z}_\Omega(\boldsymbol{x}_m)^T\boldsymbol{z}_\Omega(\boldsymbol{x}_n)$. Hence, the output associated with observation $\boldsymbol{x}_n$ can be approximated as

$$f_{n-1}(\boldsymbol{x}_n) \approx \left(\sum_{i=1}^{n-1}\alpha_i\boldsymbol{z}_\Omega(\boldsymbol{x}_i)\right)^T \boldsymbol{z}_\Omega(\boldsymbol{x}_n). \tag{7}$$

It is a matter of elementary algebra to see that (7) can be equivalently derived by approximating the system's output as $f(\boldsymbol{x}) \approx \boldsymbol{\theta}^T\boldsymbol{z}_\Omega(\boldsymbol{x})$, initializing $\boldsymbol{\theta}_0$ to $\boldsymbol{0}_D$ and iteratively applying the following gradient descent type update: $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu\varepsilon_n\boldsymbol{z}_\Omega(\boldsymbol{x}_n)$, where $\varepsilon_n = y_n - f_{n-1}(\boldsymbol{x}_n)$.

Clearly, the procedure described here, for the case of the KLMS, can be applied to any other gradient-type kernel based method, e.g., KAPSM [46], [32], NORMA [47], e.t.c. It has the advantage of modeling the solution as a fixed size vector, instead of a growing sum, a property that is quite helpful in distributed environments, as it will be discussed in section III.

III. Distributed kernel-based Learning

In this section, we discuss the problem of online learning in RKHS over distributed networks. Specifically, we consider K connected nodes, labeled $k \in \mathcal{N} = \{1, 2, \dots K\}$, which operate in cooperation with their neighbors to solve a specific task. Let $\mathcal{N}_k \subseteq \mathcal{N}$ denote the neighbors of node k. The network topology is represented as an undirected *connected* graph, consisting of K vertices (representing the nodes) and a set of edges connecting the nodes to each other (i.e., each node is connected to its neighbors). We assign a nonnegative weight $a_{k,l}$ to the edge connecting node k to l. This weight is used by k to scale the data transmitted from l and vice versa. This can be interpreted as a measure of the confidence level that node k assigns to its interaction with node l. We collect all coefficients into a $K \times K$ symmetric matrix $A = (a_{k,l})$, such that the entries of the k-th row of A contain the coefficients used by node k to scale the data arriving from its neighbors. We make the additional assumption that A is doubly stochastic, so that the weights of all incoming and outgoing "transmissions" sum to 1. A common choice, among others, for choosing these coefficients, is the *Metropolis rule*, in which the weights equal to:

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

Finally, we assume that each node, k, receives streaming data $\{(\boldsymbol{x}_{k,n}, y_{k,n}), n = 1, 2, \dots\}$, that are generated from an input-output relationship of the form $y_{k,n} = f(\boldsymbol{x}_{k,n}) + \eta_{k,n}$, where $\boldsymbol{x}_{k,n} \in \mathbb{R}^d$, $y_{k,n}$ belongs to $\mathbb{R}$ and $\eta_{k,n}$ represents the respective noise, for the regression task. The goal is to obtain an estimate of f. For classification, $y_{n,k} = \phi(f(\boldsymbol{x}_{k,n}))$, where, ϕ is a thresholding function; here we assume that $y_{n,k} \in \{-1, 1\}$. Once more, the goal is to optimally estimate the classifier function f.

Each one of the nodes aims to estimate $f \in \mathcal{H}$ by minimizing a specific convex cost function, $\mathcal{L}(\boldsymbol{x}, y, f)$, using a (sub)gradient descent approach. We employ a simple *Combine-Then-Adapt* (CTA) rationale, where at each time instant, n, each node, k, a) receives the current estimates, $f_{l,n-1}$, from all neighbors (i.e., from all nodes $l \in \mathcal{N}_k$), b) combines them to a single solution, $\psi_{k,n-1} = \sum_{l \in \mathcal{N}_k} a_{k,l} f_{l,n-1}$ and c) apply a step update procedure:

$$f_n = \psi_{k,n-1} - \mu_n \nabla_f \mathcal{L}(\boldsymbol{x}_n, y_n, \psi_{k,n-1}).$$

The implementation of such an approach in the context of RKHS presents significant challenges. Keep in mind that, the estimation of the solution at each node is not a simple vector, but instead it is a function, which is expressed as a growing sum of kernel evaluations centered at the points observed by the specific node, as in (1). Hence, the implementation of a straightforward CTA strategy would require from each node to transmit its entire growing sum (i.e., the coefficients a_i as well as the respective centers $\boldsymbol{x}_i$) to all neighbors. This would significantly increase both the communication load among the nodes, as well as the computational cost at each node, since the size of each one of the expansions would become increasingly larger as time evolves (as for every time instant, they gather the centers transmitted by all neighbors). This is the rationale adopted in [21], [22], [23] for the case of KLMS. Clearly, this is far from a practical approach. Alternatively, one could devise an efficient method to sparsify the solution at each node and then merge the sums transmitted by its neighbors. This would require (for example) to search all the dictionaries, transmitted by the neighboring nodes, for similar centers and treat them as a single one, or adopting a single pre-arranged dictionary (i.e., a specific set of centers) for all nodes and then fuse each observed point with the best-suited center. However, no such strategy has appeared in the respective literature, perhaps due to its increased complexity and lack of a theoretical elegance.

In this paper, inspired by the random Fourier features approximation technique, we approximate the desired input-output relationship as $y = \boldsymbol{\theta}^T \boldsymbol{z}_\Omega(\boldsymbol{x})$ and propose a two step procedure: a) we map each observed point $(\boldsymbol{x}_{k,n}, y_{k,n})$ to $(\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}), y_{k,n})$ and then b) we adopt a simple linear CTA diffusion strategy on the transformed points. Note that in the proposed scheme, each node aims to estimate a vector $\boldsymbol{\theta} \in \mathbb{R}^D$ by minimizing a specific (convex) cost function, $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta})$. Here, we imply that the model can be closely approximated by $y_{k,n} \approx \boldsymbol{\theta}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}) + \eta_{k,n}$, for regression, and $y_{k,n} \sim \phi(\boldsymbol{\theta}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}))$ for classification, for all k, n, for some $\boldsymbol{\theta}$. We emphasize that $\mathcal{L}$ needs not be differentiable. Hence, a large family of loss functions can be adopted. For example:

- Squared error loss: $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) = (y - \boldsymbol{\theta}^T \boldsymbol{x})^2$.
- Hinge loss: $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \max(0, 1 - y\boldsymbol{\theta}^T \boldsymbol{x})$.

We end up with the following generic update rule:

$$\boldsymbol{\psi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \boldsymbol{\theta}_{l,n-1}, \tag{8}$$

$$\boldsymbol{\theta}_{k,n} = \boldsymbol{\psi}_{k,n} - \mu_{k,n} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}), y_{k,n}, \boldsymbol{\psi}_{k,n}), \tag{9}$$

where $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}), y_{k,n}, \boldsymbol{\psi}_{k,n})$ is the gradient, or any subgradient of $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta})$ (with respect to $\boldsymbol{\theta}$), if the loss function is not differentiable. Algorithm 1 summarizes the aforementioned procedure. The advantage of the proposed scheme is that each node transmits a single vector (i.e., its current estimate, $\boldsymbol{\theta}_{k,n}$) to its neighbors, while the merging of the solutions requires only a straightforward summation.

Algorithm 1 Random Fourier Features Distributed Online Kernel-based Learning (RFF-DOKL).

```
$D = \{(\boldsymbol{x}_{k,n}, y_{k,n}), k = 1, 2 \dots, K,\ n = 1, 2, \dots\}$   ▷ Input
Select a specific shift invariant (semi)positive definite kernel, a specific loss function $\mathcal{L}$ and a sequence of possible variable learning rates $\mu_n$. Each node generates the same matrix $\Omega$ as in (6).
$\boldsymbol{\theta}_{k,0} \leftarrow \mathbf{0}_D$, for all $k$.   ▷ Initialization
for $n = 1, 2, 3, ...$ do
    for each node $k$ do
        $\boldsymbol{\psi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \boldsymbol{\theta}_{l,n-1}$.
        $\boldsymbol{\theta}_{k,n} = \boldsymbol{\psi}_{k,n} - \mu_{k,n} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}), y_{k,n}, \boldsymbol{\psi}_{k,n})$.
```

A. Consensus and regret bound

In the sequel, we will show that, under certain assumptions, the proposed scheme achieves asymptotic consensus and that the corresponding regret bound grows sublinearly with the time. It can readily be seen that (8)-(9) can be written more compactly (for the whole network) as follows:

$$\underline{\boldsymbol{\theta}}_n = \boldsymbol{A} \underline{\boldsymbol{\theta}}_{n-1} - \boldsymbol{M}_n \boldsymbol{G}_n, \tag{10}$$

where $\underline{\boldsymbol{\theta}}_n := (\boldsymbol{\theta}_{1,n}^T, \dots, \boldsymbol{\theta}_{K,n}^T)^T \in \mathbb{R}^{KD}$, $\boldsymbol{M}_n := \mathrm{diag}\{\mu_{1,n}, \dots, \mu_{K,n}\} \otimes I_D$, $\boldsymbol{G}_n := [(\boldsymbol{u}_{1,n}^T, \dots, \boldsymbol{u}_{K,n}^T]^T \in \mathbb{R}^{KD}$, where $\boldsymbol{u}_{k,n} = \nabla \mathcal{L}(\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}), y_{k,n}, \boldsymbol{\psi}_{k,n})$, and $\boldsymbol{A} := A \otimes I_D$. The necessary assumptions are the following:

Assumption 1. *The step size is time decaying and is bounded by the inverse square root of time, i.e., $\mu_{k,n} = \mu_n \leq \mu n^{-1/2}$.*

Assumption 2. *The norm of the transformed input is bounded, i.e., $\exists U_1$ such that $\|\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n})\| \leq U_1,\ \forall k \in \mathcal{N}, \forall n \in \mathbb{N}$. Furthermore, $y_{k,n}$ is bounded, i.e., $|y_{k,n}| \leq V\ \forall k \in \mathcal{N}, \forall n \in \mathbb{N}$ for some $V > 0$.*

Assumption 3. *The estimates are bounded, i.e., $\exists U_2$ s.t. $\|\boldsymbol{\theta}_{k,n}\| \leq U_2,\ \forall k \in \mathcal{N}, \forall n \in \mathbb{N}$.*

Assumption 4. *The matrix comprising the combination weights, i.e., A, is doubly stochastic (if the weights are chosen with respect to the Metropolis rule, this condition is met).*

Note that assumptions 2 and 3 (which are valid for most of the popular cost functions) guarantee that the gradient of the respective cost function is bounded (we will use this fact in the following proofs of the Appendices). As an example, we can study the squared error loss, i.e., $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) = 1/2(y - \boldsymbol{\theta}^T \boldsymbol{x})^2$, where:

$$\begin{aligned} \|\nabla \mathcal{L}(\boldsymbol{z}_\Omega(\boldsymbol{x}), y, \boldsymbol{\theta})\| &\leq |y| \|\boldsymbol{z}_\Omega(\boldsymbol{x})\| + \|\boldsymbol{\theta}\| \|\boldsymbol{z}_\Omega(\boldsymbol{x})\|^2 \\ &\leq V U_1 + U_1^2 U_2. \end{aligned}$$

Following similar arguments, we can also prove that many other popular cost functions (e.g., the hinge loss, the logistic loss, e.t.c.) have bounded gradients too.

Proposition 1 (Asymptotic Consensus)**.** *All nodes converge to the same solution.*

Proof. Consider a $KD \times KD$ *consensus* matrix $\boldsymbol{A}$ as in (10). As $\boldsymbol{A}$ is doubly stochastic, we have the following [9]:

- $\|\boldsymbol{A}\| = 1$.
- Any consensus matrix $\boldsymbol{A}$ can be decomposed as
$$\boldsymbol{A} = \boldsymbol{X} + \boldsymbol{B}\boldsymbol{B}^T, \tag{11}$$
where $\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_D]$ is an $KD \times D$ matrix, and $\boldsymbol{b}_k = 1/\sqrt{K}(\boldsymbol{1} \otimes \boldsymbol{e}_k)$, where $\boldsymbol{e}_k$, $k = 1, \dots, D$ represent the standard basis of $\mathbb{R}^D$ and $\boldsymbol{X}$ is a $KD \times KD$ matrix for which it holds that $\|\boldsymbol{X}\| < 1$.
- $\boldsymbol{A}\underline{\breve{\boldsymbol{\theta}}} = \underline{\breve{\boldsymbol{\theta}}}$, for all $\underline{\breve{\boldsymbol{\theta}}} \in \mathcal{O} := \{\underline{\boldsymbol{\theta}} \in \mathbb{R}^{KD} : \underline{\boldsymbol{\theta}} = [\boldsymbol{\theta}^T, \dots, \boldsymbol{\theta}^T]^T,\ \boldsymbol{\theta} \in \mathbb{R}^D\}$. The subspace $\mathcal{O}$ is the so called consensus subspace of dimension D, and $\boldsymbol{b}_k,\ k = 1, \dots, D$, constitute a basis for this space. Hence, the orthogonal projection of a vector, $\underline{\boldsymbol{\theta}}$, onto this linear subspace is given by $P_{\mathcal{O}}(\underline{\boldsymbol{\theta}}) := \boldsymbol{B}\boldsymbol{B}^T\underline{\boldsymbol{\theta}}$, for all $\underline{\boldsymbol{\theta}} \in \mathbb{R}^{KD}$.

In [9], it has been proved that, the algorithmic scheme achieves asymptotic consensus, i.e., $\|\boldsymbol{\theta}_{k,n} - \boldsymbol{\theta}_{l,n}\| \to 0$, as $n \to \infty$, for all $k, l \in \mathcal{N}$, if and only if $\lim_{n\to\infty} \|\underline{\boldsymbol{\theta}}_n - P_{\mathcal{O}}(\underline{\boldsymbol{\theta}}_n)\| = 0$. We can easily check that the quantity

$$\underline{\boldsymbol{r}}_n := \underline{\boldsymbol{\theta}}_{n+1} - \boldsymbol{A}\underline{\boldsymbol{\theta}}_n = -\boldsymbol{M}_{n+1}\boldsymbol{G}_{n+1}. \tag{12}$$

approaches 0, as $n \to \infty$, since $\lim_{n\to\infty} \boldsymbol{M}_n = \boldsymbol{O}_{KD}$ (assumption 1) and the matrix $\boldsymbol{G}_n$ is bounded for all n. Rearranging the terms of (12) and iterating over n, we have:

$$\begin{aligned}\underline{\boldsymbol{\theta}}_{n+1} &= \boldsymbol{A}\underline{\boldsymbol{\theta}}_n + \underline{\boldsymbol{r}}_n = \boldsymbol{A}\boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1} + \boldsymbol{A}\underline{\boldsymbol{r}}_{n-1} + \underline{\boldsymbol{r}}_n = \dots \\ &= \boldsymbol{A}^{n+1}\underline{\boldsymbol{\theta}}_0 + \sum_{j=0}^{n} \boldsymbol{A}^{n-j}\underline{\boldsymbol{r}}_j.\end{aligned}$$

If we left-multiply the previous equation by $(\boldsymbol{I}_{KD} - \boldsymbol{B}\boldsymbol{B}^T)$ and follow similar steps as in [9, Lemma 2], it can be verified that $\lim_{n\to\infty} \| \left(\boldsymbol{I}_{Km} - \boldsymbol{B}\boldsymbol{B}^T\right) \underline{\boldsymbol{\theta}}_{n+1}\| = 0$, which completes our proof. $\square$

Proposition 2. *Under assumptions 1-4 (and a cost function with bounded gradients) the networkwise regret is bounded by*

$$\sum_{i=1}^{N} \sum_{k\in\mathcal{N}} (\mathcal{L}(\boldsymbol{x}_{k,i}, y_{k,i}, \boldsymbol{\psi}_{k,i}) - \mathcal{L}(\boldsymbol{x}_{k,i}, y_{k,i}, \boldsymbol{g})) \le \gamma\sqrt{N} + \delta,$$

for all $\boldsymbol{g} \in \mathcal{B}_{[\boldsymbol{0}_D, U_2]}$, *where* γ, δ *are positive constants and* $\mathcal{B}_{[\boldsymbol{0}_D, U_2]}$ *is the closed ball with center* $\boldsymbol{0}_D$ *and radius* U_2.

Proof. See appendix A. $\square$

Remark 1. *It is worth pointing out that the theoretical properties, which were stated before, are complementary. In particular, the consensus property (Proposition 1) indicates that the nodes converge to the* ***same*** *solution and the sub-linearity of the regret implies that on average the algorithm performs as well as the best fixed strategy. In fact, without the regret related proof we cannot characterize the solution in which the nodes converge.*

B. Diffusion SVM (Pegasos) Algorithm

The case of the regularized hinge loss function, i.e., $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 + \max\{0, 1 - y\boldsymbol{\theta}^T\boldsymbol{z}_\Omega(\boldsymbol{x})\}$, for a specific value of the regularization parameter $\lambda > 0$, generates the *Distributed Pegasos* (see [34]). Note that the Pegasos solves the SVM task in the primal domain. In this case, the gradient becomes $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \lambda\boldsymbol{\theta} - \boldsymbol{I}_+(1 - y\boldsymbol{\theta}^T\boldsymbol{z}_\Omega(\boldsymbol{x}))y\boldsymbol{z}_\Omega(\boldsymbol{x})$, where $\boldsymbol{I}_+$ is the indicator function of $(0, +\infty)$, which takes a value of 1, if its argument belongs in $(0, +\infty)$, and zero otherwise. Hence the step-update equation of algorithm 1 becomes:

$$\begin{aligned}\boldsymbol{\theta}_{k,n} = \ & (1 - \tfrac{1}{n})\boldsymbol{\psi}_{k,n} \\ & + \boldsymbol{I}_+(1 - y_n\boldsymbol{\psi}_{k,n}^T\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}))\tfrac{y_{k,n}}{\lambda n}\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}),\end{aligned} \tag{13}$$

where, following [34], we have used a decreasing step size, $\mu_n = \frac{1}{\lambda n}$. This scheme satisfies the required assumptions, hence consensus is guaranteed.

We have tested the performance of Distributed-Pegasos versus the non-cooperative Pegasus on four datasets downloaded from Leon Bottou's LASVM web page [48]. The chosen datasets are: a) the Adult dataset, b) the Banana dataset (where we have used the first 4000 points as training data and the remaining 1300 as testing data), c) the Waveform dataset (where we have used the first 4000 points as training data and the remaining 1000 as testing data) and d) the MNIST dataset (for the task of classifying the digit 8 versus the rest). The sizes of the datasets are given in Table I. In all experiments, we generate random graphs (using MIT's random_graph routine, see [49]) and compare the proposed diffusion method versus a noncooperative strategy (where each node works independent of the rest). For each realization of the experiments, a different random connected graph with $K = 5$ or $K = 20$ nodes was generated, with probability of attachment per node equal to 0.2 (i.e, there is a 20% probability that a specific node k is connected to any other node l). The adjacency matrix, A, of each graph was generated using the Metropolis rule. For the non-cooperative strategies, we used a graph that connects each node to itself, i.e., $A = I_5$ or $A = I_{20}$ respectively. The latter, implies that no information is exchanged between the nodes, thus each node is working alone. Moreover, for each realization, the corresponding dataset was randomly split into K subsets of equal size (one for every node).

We note that the value of D affects significantly the quality of the approximation via the Fourier features rationale and thus it also affects the performance of the experiments. The value of D must be large enough so that the approximation is good, but not too large so that to the communicational and computational load become affordable. In practice, we can find a value for D (after trials) so that any further increase results to almost negligible performance variation (see also section IV). All other parameters were optimized (after trials) to give the lowest number of training errors. Their values are reported on Table IV. The algorithms were implemented in MatLab and the experiments were performed on a i7-3770 machine running at 3.4GHz with 32 Mb of RAM. Tables II and III report the mean test errors (and the standard deviations) obtained by both procedures. For $K = 5$, the mean algebraic complexity of the generated graphs lies between 0.61 and

TABLE I
DATASET INFORMATION.

Method	Adult	Banana	Waveform	MNIST
Training size	32562	4000	4000	60000
Testing size	16282	1300	1000	10000
dimensions	123	2	21	784

0.76 (different for each experiment), while the corresponding mean algebraic degree lies around 1.8. For $K = 20$, the mean algebraic complexity of the generated graphs lies around 0.70, while the corresponding mean algebraic degree lies around 3.9. The number inside the parentheses indicates the times of data reuse (i.e., running the algorithm again over the same data, albeit with a continuously decreasing step-size μ_n), which has been suggested that improves the classification accuracy of Pegasos (see [34]). For example, the number 2 indicates that the algorithm runs over a dataset of double size, that contains the same data pairs twice. For the three first datasets (Adult, Banana, Waveform) we have run 100 realizations of the experiment, while for the fourth (MNIST) we have run only 10 (to save time). In that particular dataset we have used a rather large D, thus the training requires significantly more time compared to the other datasets. Besides the ADULT dataset, all other simulations show that the distributed implementation significantly outperforms the non-cooperative one. For that particular dataset, we observe that for a single run the non-cooperative strategy behaves better (for $K = 20$), but as data reuse increases the distributed implementation reaches lower error floors.

C. Diffusion KLMS

Adopting the mean squared error in place of $\mathcal{L}$, i.e., $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) = E[(y - \boldsymbol{\theta}^T \boldsymbol{z}_\Omega(\boldsymbol{x}))^2]$, and estimating the gradient by its current measurement, we take the Random Fourier Features Diffusion KLMS (RFF-DKLMS) and the step update becomes:

$$\boldsymbol{\theta}_{k,n} = \boldsymbol{\psi}_{k,n-1} + \mu \varepsilon_{k,n} \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}), \tag{14}$$

where $\varepsilon_{k,n} = y_n - \boldsymbol{\psi}_{k,n-1}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n})$. Although proposition 1 cannot be applied here (as it requires a decreasing step-size), we can derive sufficient conditions for consensus following the results of the standard Diffusion LMS [8]. Henceforth, we will assume that the data pairs are generated by

$$y_{k,n} = \sum_{m=1}^{M} a_m \kappa(\boldsymbol{c}_m, \boldsymbol{x}_{k,n}) + \eta_{k,n}, \tag{15}$$

where $\boldsymbol{c}_1, \dots, \boldsymbol{c}_M$ are fixed centers, $\boldsymbol{x}_{k,n}$ are zero-mean i.i.d, samples drawn from the Gaussian distribution with covariance matrix $\sigma_x^2 \boldsymbol{I}_d$ and $\eta_{k,n}$ are i.i.d. noise samples drawn from $\mathcal{N}(0, \sigma_\eta^2)$. Following the RFF approximation rationale (for shift invariant kernels), we can write that

$$\begin{aligned} y_{k,n} &= \sum_{m=1}^{M} a_m E_{\boldsymbol{\omega},\boldsymbol{b}}[z_{\boldsymbol{\omega},\boldsymbol{b}}(\boldsymbol{c}_m) z_{\boldsymbol{\omega},\boldsymbol{b}}(\boldsymbol{x}_{k,n})] + \eta_{k,n} \\ &= \boldsymbol{a}^T Z_\Omega^T \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}) + \epsilon_{k,n} + \eta_{k,n}, \\ &= \boldsymbol{\theta}_o^T \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}) + \epsilon_{k,n} + \eta_{k,n}, \end{aligned}$$

where $Z_\Omega = (\boldsymbol{z}_\Omega(\boldsymbol{c}_1), \dots, \boldsymbol{z}_\Omega(\boldsymbol{c}_M))$, $\boldsymbol{a} = (a_1, \dots, a_M)^T$, $\boldsymbol{\theta}_o = Z_\Omega \boldsymbol{a}$ and $\epsilon_{k,n}$ is the approximation error between the noise-free component of $y_{k,n}$ (evaluated only by the linear kernel expansion of (15)) and the approximation of this component using random Fourier features, i.e., $\epsilon_{k,n} = \sum_{m=1}^{M} a_m \kappa(\boldsymbol{c}_m, \boldsymbol{x}_{k,n}) - \boldsymbol{\theta}_o^T \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n})$. For the whole network we have the following

$$\underline{\boldsymbol{y}}_n = \boldsymbol{V}_n^T \underline{\boldsymbol{\theta}}_o + \underline{\boldsymbol{\epsilon}}_n + \underline{\boldsymbol{\eta}}_n, \tag{16}$$

where

- $\underline{\boldsymbol{y}}_n := (y_{1,n}, y_{2,n}, \dots, y_{K,n})^T$,
- $\boldsymbol{V}_n := \operatorname{diag}(\boldsymbol{z}_\Omega(\boldsymbol{x}_{1,n}), \boldsymbol{z}_\Omega(\boldsymbol{x}_{2,n}), \dots, \boldsymbol{z}_\Omega(\boldsymbol{x}_{K,n}))$, is a $DK \times K$ matrix,
- $\underline{\boldsymbol{\theta}}_o = \left(\boldsymbol{\theta}_o^T, \boldsymbol{\theta}_o^T, \dots, \boldsymbol{\theta}_o^T\right)^T \in \mathbb{R}^{DK}$,
- $\underline{\boldsymbol{\epsilon}}_n = (\epsilon_{1,n}, \epsilon_{2,n}, \dots, \epsilon_{K,n})^T \in \mathbb{R}^K$,
- $\underline{\boldsymbol{\eta}}_n = (\eta_{1,n}, \eta_{2,n}, \dots, \eta_{K,n})^T \in \mathbb{R}^K$.

Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_K \in \mathbb{R}^d$, $\underline{\boldsymbol{y}} \in \mathbb{R}^K$, be the random variables that generate the measurements of the nodes; it is straightforward to prove that the corresponding Wiener solution, i.e., $\underline{\boldsymbol{\theta}}_* = \operatorname{argmin}_{\underline{\boldsymbol{\theta}}} E[\|\underline{\boldsymbol{y}} - \boldsymbol{V}^T \underline{\boldsymbol{\theta}}\|^2]$, becomes

$$\underline{\boldsymbol{\theta}}_* = E[\boldsymbol{V}\boldsymbol{V}^T]^{-1} E[\boldsymbol{V}\underline{\boldsymbol{y}}], \tag{17}$$

provided that the autocorrelation matrix $\boldsymbol{R} = E[\boldsymbol{V}\boldsymbol{V}^T]$ is invertible, where $\boldsymbol{V} = \operatorname{diag}(\boldsymbol{z}_\Omega(\boldsymbol{x}_1), \boldsymbol{z}_\Omega(\boldsymbol{x}_2), \dots, \boldsymbol{z}_\Omega(\boldsymbol{x}_K))$ is a $DK \times K$ matrix that collects the transformed random variables for the whole network. Assuming that the input-output relationship of the measurements at each node follows (16), the cross-correlation vector takes the form

$$\begin{aligned} E[\boldsymbol{V}\underline{\boldsymbol{y}}] &= E[\boldsymbol{V}(\boldsymbol{V}^T \underline{\boldsymbol{\theta}}_o + \underline{\boldsymbol{\epsilon}} + \underline{\boldsymbol{\eta}})] \\ &= E[\boldsymbol{V}\boldsymbol{V}^T]\underline{\boldsymbol{\theta}}_o + E[\boldsymbol{V}\underline{\boldsymbol{\epsilon}}], \end{aligned}$$

where for the last relation we have used that $\boldsymbol{\eta}$ is a zero mean vector representing noise and that $\boldsymbol{V}$ and $\underline{\boldsymbol{\eta}}$ are independent. For large enough D, the approximation error vector $\underline{\boldsymbol{\epsilon}}$ approaches $\boldsymbol{0}_K$ [43], [44], hence the optimal solution becomes:

$$\begin{aligned} \underline{\boldsymbol{\theta}}_* &= E[\boldsymbol{V}\boldsymbol{V}^T]^{-1}\left(E[\boldsymbol{V}\boldsymbol{V}^T]\underline{\boldsymbol{\theta}}_o + E[\boldsymbol{V}\underline{\boldsymbol{\epsilon}}]\right) \\ &= \underline{\boldsymbol{\theta}}_o + E[\boldsymbol{V}\boldsymbol{V}^T]^{-1} E[\boldsymbol{V}\underline{\boldsymbol{\epsilon}}] \approx \underline{\boldsymbol{\theta}}_o. \end{aligned}$$

Here we actually imply that (16) can be closely approximated by $\boldsymbol{y}_n \approx \boldsymbol{V}_n \underline{\boldsymbol{\theta}}_o + \underline{\boldsymbol{\eta}}_n$; hence, the RFF-DKLMS is actually the standard diffusion LMS applied on the data pairs $\{(\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n}), y_{k,n}),\ k = 1, \dots, K,\ n = 1, 2 \dots\}$. The difference is that the input vectors $\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n})$ may have non zero mean and do not follow, necessarily, the Gaussian distribution. Hence, the available results regarding convergence and stability of diffusion LMS (e.g., [50], [51]) cannot be applied directly (in these works the inputs are assumed to be zero mean Gaussian to simplify the formulas related to stability). To this end, we will follow a slightly different approach. Regarding the autocorrelation matrix, we have the following result:

Lemma 1. *Consider a selection of samples $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \dots, \boldsymbol{\omega}_D$, drawn from (4) such that $\boldsymbol{\omega}_i \neq \boldsymbol{\omega}_j$, for any $i \neq j$. Then, the matrix $\boldsymbol{R} = E[\boldsymbol{V}\boldsymbol{V}^T]$ is strictly positive definite (hence invertible).*

TABLE II
COMPARING THE PERFORMANCES OF THE DISTRIBUTED PEGASOS VERSUS THE NON-COOPERATIVE PEGASOS FOR GRAPHS WITH $K = 5$ NODES.

Method	Adult	Banana	Waveform	MNIST
Distributed-Pegasos (1)	19.29±2.34%	12.00±1.23%	12.3±1.51%	0.79±0.11%
Distributed-Pegasos (2)	17.84±2.01%	10.81±0.60%	10.61±0.95%	0.68±0.06%
Distributed-Pegasos (5)	15.89±0.62%	10.32±0.42%	9.61±0.63%	0.56±0.03%
Non-cooperative-Pegasos (1)	19.47±1.53%	14.30±1.39%	13.65±1.26%	1.49±0.11%
Non-cooperative-Pegasos (2)	18.67±1.42%	12.46±0.61%	12.57±0.83%	1.13±0.05%
Non-cooperative-Pegasos (5)	17.46±0.84%	11.24±0.42%	11.87±0.51%	1.03±0.03%

TABLE III
COMPARING THE PERFORMANCES OF THE DISTRIBUTED PEGASOS VERSUS THE NON-COOPERATIVE PEGASOS FOR GRAPHS WITH $K = 20$ NODES.

Method	Adult	Banana	Waveform	MNIST
Distributed-Pegasos (1)	23.60±3.67%	16.58±1.54%	16.37±1.82%	0.95±0.18%
Distributed-Pegasos (2)	21.95±2.45%	13.04±0.94%	13.53±1.35%	0.75±0.06%
Distributed-Pegasos (5)	18.45±1.24%	10.79±0.41%	11.18±0.79%	0.56±0.05%
Non-cooperative-Pegasos (1)	20.77±1.15%	21.50±1.45%	18.48±0.98%	2.95±0.16%
Non-cooperative-Pegasos (2)	20.44±0.89%	18.50±1.06%	16.50±0.72%	2.19±0.03%
Non-cooperative-Pegasos (5)	19.86±0.65%	15.95±0.76%	14.92±0.52%	1.89±0.03%

TABLE IV
PARAMETERS FOR EACH METHOD.

Method	Adult	Banana	Waveform	MNIST
Kernel-Pegasos	$\sigma = \sqrt{10}$ $\lambda = 0.0000307$	$\sigma = 0.7$ $\lambda = \frac{1}{316}$	$\sigma = \sqrt{10}$ $\lambda = 0.001$	$\sigma = 4$ $\lambda = 10^{-7}$
RFF-Pegasos	$\sigma = \sqrt{10}$ $\lambda = 0.0000307$ $D = 2000$	$\sigma = 0.7$ $\lambda = \frac{1}{316}$ $D = 200$	$\sigma = \sqrt{10}$ $\lambda = 0.001$ $D = 2000$	$\sigma = 4$ $\lambda = 10^{-7}$ $D = 100000$

Proof. Observe that the $DK \times DK$ autocorrelation matrix is given by $\boldsymbol{R} = E[\boldsymbol{V}\boldsymbol{V}^T] = \mathrm{diag}(R_{zz}, R_{zz} \dots, R_{zz})$, where $R_{zz} = E[\boldsymbol{z}_\Omega(\boldsymbol{x}_k)\boldsymbol{z}_\Omega(\boldsymbol{x}_k)^T]$, for all $k = 1, 2, \dots, K$. It suffices to prove that the $D \times D$ matrix R_{zz} is strictly positive definite. Evidently, $\boldsymbol{c}^T R_{zz}\boldsymbol{c} = \boldsymbol{c}^T E\left[\boldsymbol{z}_\Omega(\boldsymbol{x}_k)\boldsymbol{z}_\Omega(\boldsymbol{x}_k)^T\right]\boldsymbol{c} = E\left[\left(\boldsymbol{z}_\Omega(\boldsymbol{x}_k)^T\boldsymbol{c}\right)^2\right] \geq 0$, for all $\boldsymbol{c} \in \mathbb{R}^D$. Now, assume that there is a $\boldsymbol{c} \in \mathbb{R}^D$ such that $E\left[\left(\boldsymbol{z}_\Omega(\boldsymbol{x}_k)^T\boldsymbol{c}\right)^2\right] = 0$. Then $\boldsymbol{z}_\Omega(\boldsymbol{x})^T\boldsymbol{c} = 0$ for all $\boldsymbol{x} \in \mathbb{R}^D$, or equivalently, $\sum_{i=1}^D c_i \cos(\boldsymbol{\omega}_i^T\boldsymbol{x} + b_i) = 0$, for all $\boldsymbol{x} \in \mathbb{R}^D$. Thus, $\boldsymbol{c} = \boldsymbol{0}$. □

As expected, the eigenvalues of R_{zz} play a pivotal role in the convergence's study of the algorithm. As R_{zz} is a strictly positive definite matrix, its eigenvalues satisfy $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_D$.

Proposition 3. *If the the step update μ satisfies: $0 < \mu < \frac{2}{\lambda_D}$, where λ_D is the maximum eigenvalue of R_{zz}, then the RFF-DKLMS achieves asymptotic consensus in the mean, i.e.,*

$$\lim_n E[\boldsymbol{\theta}_{k,n} - \boldsymbol{\theta}_o] = \boldsymbol{0}_D, \text{ for all } k = 1, 2, \dots, K.$$

Proof. See Appendix B. □

Remark 2. *If $\boldsymbol{x}_{k,n} \sim \mathcal{N}(\boldsymbol{0}, \sigma_X \boldsymbol{I}_d)$, it is possible to evaluate explicitly the entries of R_{zz}, i.e.,*

$$\begin{aligned} r_{i,j} =& \frac{1}{2}\exp\left(\frac{-\|\boldsymbol{\omega}_i - \boldsymbol{\omega}_j\|^2\sigma_X^2}{2}\right)\cos(b_i - b_j) \\ &+ \frac{1}{2}\exp\left(\frac{-\|\boldsymbol{\omega}_i + \boldsymbol{\omega}_j\|^2\sigma_X^2}{2}\right)\cos(b_i + b_j). \end{aligned}$$

Note that up to this point, we have not used the Gaussian property of $\boldsymbol{x}_{k,n}$. We use it here to compute the elements of R_{zz}, which plays a significant role in the stability condition that follows.

Proposition 4. *For stability in the mean-square sense, we must ensure that both μ and A satisfy:*

$$|\rho\left(\boldsymbol{I}_{D^2K^2} - \mu\left(\boldsymbol{R} \boxtimes \boldsymbol{I}_{DK} + \boldsymbol{I}_{DK} \boxtimes \boldsymbol{R}\right)\left(\boldsymbol{A} \boxtimes \boldsymbol{A}\right)\right)| < 1,$$

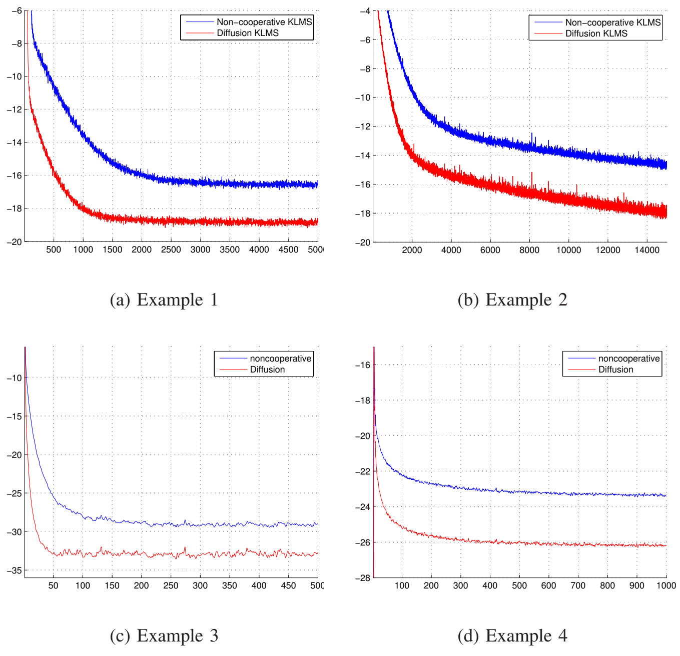
where $\boxtimes$ denotes the unbalanced block Kronecker product and ρ the spectral radius.

Proof. See Appendix C. □

In the following, we present some experiments to illustrate the performance of the proposed scheme. We demonstrate that the estimation provided by the cooperative strategy is better than having each node working alone (i.e., lower MSE). Similar to section III-C, each realization of the experiments uses a different random connected graph with $M = 20$ nodes and probability of attachment per node equal to 0.2. The adjacency matrix, A, of each graph was generated using the Metropolis rule (resulting to graphs with mean algebraic connectivity around 0.69), while for the non-cooperative strategies, we used a graph that connects each node to itself, i.e., $A = I_{20}$. All parameters were optimized (after trials) to give the lowest MSE. The algorithms were implemented in MatLab and the experiments were performed on a i7-3770 machine running at 3.4GHz with 32 Mb of RAM.

1) Example 1. A linear expansion in terms of kernels: In this set-up, we generate 5000 data pairs for each node using the following model: $y_{k,n} = \sum_{m=1}^{500} a_m \kappa(\boldsymbol{c}_m, \boldsymbol{x}_{k,n}) + \eta_{k,n}$, where $\boldsymbol{x}_{k,n} \in \mathbb{R}^5$ are drawn from $\mathcal{N}(\boldsymbol{0}, I_5)$ and the noise are i.i.d. Gaussian samples with $\sigma_\eta = 0.1$. The parameters of the expansion (i.e., $a_1, \dots, a_M$) are drawn from $\mathcal{N}(0, 25)$, the kernel parameter σ is set to 5, the step update to $\mu = 1$ and the number of random Fourier features to $D = 2500$. Figure 1(a) shows the evolution of the MSE over all network nodes for 100 realizations of the experiment. We note that the selected value of step size satisfies the conditions of proposition 3.

2) Example 2: Next, we generate the data pairs for each node using the following simple non-linear model: $y_{k,n} = \boldsymbol{w}_0^T\boldsymbol{x}_{k,n} + 0.1 \cdot (\boldsymbol{w}_1^T\boldsymbol{x}_{k,n})^2 + \eta_{k,n}$, where $\eta_{k,n}$ represent zero-mean i.i.d. Gaussian noise with $\sigma_\eta = 0.05$ and the coefficients



(a) Example 1 (b) Example 2

(c) Example 3 (d) Example 4

Fig. 1. Comparing the performances of RFF Diffusion KLMS versus the non-cooperative strategy.

of the vectors $\boldsymbol{w}_0, \boldsymbol{w}_1 \in \mathbb{R}^5$ are i.i.d. samples drawn from $\mathcal{N}(0,1)$. Similarly to Example 1, the kernel parameter σ is set to 5 and the step update to $\mu = 1$. The number of random Fourier coefficients for RFF-DKLMS was set to $D = 300$. Figure 3(b) shows the evolution of the MSE over all network nodes for 1000 realizations of the experiment over 15000 samples.

3) Example 3: Here we adopt the following chaotic series model [52]: $d_{k,n} = \frac{d_{k,n-1}}{1+d^2_{k,n-1}} + u^3_{k,n-1}$, $y_{k,n} = d_{k,n} + \eta_{k,n}$, where η_n is zero-mean i.i.d. Gaussian noise with $\sigma_\eta = 0.01$ and u_n is also zero-mean i.i.d. Gaussian with $\sigma_u = 0.15$. The kernel parameter σ is set to 0.05, the number of Fourier features to $D = 100$ and the step update to $\mu = 1$. We have also initialized d_1 to 1. Figure 1(c) shows the evolution of the MSE over all network nodes for 1000 realizations of the experiment over 500 samples.

4) Example 4: For the final example, we use another chaotic series model [52]: $d_{k,n} = u_{k,n} + 0.5v_{k,n} - 0.2d_{k,n-1} + 0.35d_{k,n-2}$, $y_{k,n} = \phi(d_{k,n}) + \eta_{k,n}$,

$$\phi(d_{k,n}) = \begin{cases} \frac{d_{k,n}}{3(0.1+0.9d^2_{k,n})^{1/2}} & d_{k,n} \geq 0 \\ \frac{-d^2_{k,n}(1-\exp(0.7d_{k,n}))}{3} & d_{k,n} < 0 \end{cases},$$

where $\eta_{k,n}$, $v_{k,n}$ are zero-mean i.i.d. Gaussian noise with $\sigma_\eta = 0.001$ and $\sigma_v^2 = 0.0156$ respectively, and $u_{k,n} = 0.5v_{k,n} + \hat{\eta}_{k,n}$, where $\hat{\eta}_n$ is also i.i.d. Gaussian with $\sigma^2 = 0.0156$. The kernel parameter σ is set to 0.05 and the step update to $\mu = 1$. We have also initialized d_1, d_2 to 1. Figure 3(d) shows the evolution of the MSE over all network nodes for 1000 realizations of the experiment over 1000 samples. The number of random Fourier features was set to $D = 200$.

D. Computational Load Saving

In section III, we discussed the problems arising from a straightforward "naive" implementation of a diffusion strategy in kernel-based methods, as in [21], [22], [23]. In this section we explicitly show how the proposed scheme can significantly reduce the impractical computational requirements of these methods. Consider a simple network topology of K nodes, where the probability of attachment per node is p. This practically means that each node is connected to pK nodes on average. Furthermore, we assume that each node, k, receives streaming data of the form $\{(x_{k,n}, y_{k,n}), n = 1, 2, ...\}$. Now consider a standard CTA diffusion strategy applied on this typical setting for the KLMS. Initially, at iteration $n = 1$, each one of the nodes will store the first arriving center to its respective dictionary (the set containing the coefficients and the centers of the expansion of the estimated solution), i.e., node k will store $x_{k,1}$. At the second iteration, each node k will gather the dictionaries of all neighboring nodes together with the currently observed center $x_{k,2}$. Thus, each center will store $pK + 1$ centers on average. Subsequently, at the third iteration each node will transmit $pK + 1$ centers and gather $pK(pK + 1) + 1$ centers on average. It is not difficult to see that at iteration n each node will gather (on average)

$$(pK)^{n-1} + (pK)^{n-2} + \cdots + 1 = \frac{(pK)^n - 1}{pK - 1}$$

centers. This means that each node will have to transmit $\frac{(pK)^{n-1}-1}{pK-1}$ centers and store $\frac{(pK)^n-1}{pK-1}(d+1)$ floating point numbers (the centers and the coefficients of the current expansion). Furthermore, in order to compute the estimated solution, each node would also require $\frac{(pK)^n-1}{pK-1}$ kernel evaluations. Clearly, this is impractical.

On the other hand, the proposed method requires the storage of the matrix Ω and the estimated solution $\boldsymbol{\theta}_{k,n} \in \mathbb{R}^D$, thus $(d+2)D$ floating point numbers in total, while in order to compute the estimated solution, the node would require $\mathcal{O}(dD)$ floating point evaluations. Therefore, at iteration n the proposed scheme has $\mathcal{O}\left(\frac{(pK)^{n-1}}{D}\right)$ times lower storage/computational requirements. Furthermore, the communication load of the network is reduced by a factor proportional to $\frac{(pK)^{n-2}}{D}$. As we can see the savings increase exponentially over time.

IV. Revisiting Online Kernel based learning

In this section, we investigate the use of random Fourier features as a general framework for online kernel-based learning. The framework presented here can be seen as a special case of the general distributed method presented in section III for a network with a single node. Similar to the case of the standard KLMS, the learning algorithms considered here adopt a gradient descent rationale to minimize a specific loss function, $\mathcal{L}(\boldsymbol{x}, y, f)$ for $f \in \mathcal{H}$, so that f approximates the relationship between $\boldsymbol{x}$ and y, where $\mathcal{H}$ is the RKHS induced by a specific choice of a shift invariant (semi)positive definite kernel, κ. Hence, in general, these algorithms can be summarized by the following step update equation: $f_n = f_{n-1} + \mu_n \nabla_f \mathcal{L}(\boldsymbol{x}_n, y_n, f_{n-1})$. Algorithm 2 summarizes the

Algorithm 2 Random Fourier Features Online Kernel-based Learning (RFF-OKL).

```
$D = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \dots\}$                    ▷ Input
Select a specific (semi)positive definite kernel, a specific loss function $\mathcal{L}$ and a sequence of possible variable learning rates $\mu_n$. Then generate the matrix $\Omega$ as in (6).
$\boldsymbol{\theta}_0 \leftarrow \mathbf{0}_D$                    ▷ Initialization
for $n = 1, 2, 3, ...$ do
    $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}_{n-1})$.    ▷ Step update
```

proposed procedure for online kernel-based learning. The performance of the algorithm depends on the quality of the adopted approximation. Hence, a sufficiently large D has to be selected.

Although algorithm 2 is given in a general setting, in the following we focus on the fixed-budget KLMS. As it has been discussed in section II, KLMS adopts the MSE cost function, which in the proposed framework takes the form: $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) = E[(y_n - \boldsymbol{\theta}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_n))^2]$. Hence, the respective step update equation of algorithm 2 becomes

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu \varepsilon_n \boldsymbol{z}_\Omega(\boldsymbol{x}_n), \tag{18}$$

where $\varepsilon_n = y_n - \boldsymbol{\theta}_{n-1}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_n)$. Observe that, contrary to the typical implementations of KLMS, where the system's output is a growing expansion of kernel functions and hence special care has to be carried out to prune the so called dictionary, the proposed approach employs a fixed-budget rationale, which doesn't require any further treatment. We call this scheme the Random Fourier Features KLMS (RFF-KLMS) [53], [54].

The study of the convergence properties of RFFKLMS is based on those of the standard LMS. Henceforth, we will assume that the data pairs are generated by

$$y_n = \sum_{m=1}^{M} a_m \kappa(\boldsymbol{c}_m, \boldsymbol{x}_n) + \eta_n, \tag{19}$$

where $\boldsymbol{c}_1, \dots, \boldsymbol{c}_M$ are fixed centers, $\boldsymbol{x}_n$ are zero-mean i.i.d, samples drawn from the Gaussian distribution with covariance matrix $\sigma_x^2 \boldsymbol{I}_d$ and η_n are i.i.d. noise samples drawn from $\mathcal{N}(0, \sigma_\eta^2)$. Similar to the diffusion case, the eigenvalues of R_{zz}, i.e., $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_D$, play a pivotal role in the convergence's study of the algorithm. Applying similar assumptions as in the case of the standard LMS (e.g., independence between $\boldsymbol{x}_n, \boldsymbol{x}_m$, for $n \neq m$ and between $\boldsymbol{x}_n, \eta_n$), we can prove the following results.

Proposition 5. *For datasets generated by* (19) *we have:*

1) *If $0 < \mu < 2/\lambda_D$, then RFFKLMS converges in the mean, i.e., $E[\boldsymbol{\theta}_n - \boldsymbol{\theta}_o] \rightarrow \mathbf{0}$.*
2) *The optimal MSE is given by*

$$\mathcal{L}_n^{opt} = \sigma_\eta^2 + E[\epsilon_n^2] - E[\epsilon_n \boldsymbol{z}_\Omega(\boldsymbol{x}_n)] R_{zz}^{-1} E[\epsilon_n \boldsymbol{z}_\Omega(\boldsymbol{x}_n)^T].$$

For large enough D, we have $\mathcal{L}_n^{opt} \approx \sigma_\eta^2$.

3) *The excess MSE is given by $\mathcal{L}_n^{ex} = \mathcal{L}_n - \mathcal{L}_n^{opt} = \operatorname{tr}(R_{zz} A_n)$, where $A_n = E[(\boldsymbol{\theta}_n - \boldsymbol{\theta}_o)(\boldsymbol{\theta}_n - \boldsymbol{\theta}_o)^T]$, where $\mathcal{L}_n$ is the MSE at step n.*
4) *If $0 < \mu < 1/\lambda_D$, then A_n converges. For large enough n and D we can approximate A_n's evolution as $A_{n+1} \approx A_n - \mu(R_{zz} A_n + A_n R_{zz}) + \mu^2 \sigma_\eta^2 R_{zz}$. Using this model we can roughly approximate the steady-state MSE ($\approx \operatorname{tr}(R_{zz} A_n) + \sigma_\eta^2$).*

Proof. The proofs use standard arguments as in the case of the standard LMS. Hence we do not provide full details. The reader is addressed to any LMS textbook. The main difference to the standard LMS approach is that in this case we cannot exploit the statistics of the inputs to simplify the model of A_n. Instead, we can employ the more general rationale of zeroth-order approximation near convergence [55].
1) See Proposition 3.
2) Replacing $\boldsymbol{\theta}_n$ with $\boldsymbol{\theta}_{\text{o}}$ in $\mathcal{L}_n = E[\varepsilon_n^2]$ gives the result. For large enough D, ϵ_n is almost zero, hence we have $\mathcal{L}_n^{\text{opt}} \approx \sigma_\eta^2$.
3) Here, we use the additional assumptions that $\boldsymbol{v}_n$ is independent of $\boldsymbol{x}_n$ and that ϵ_n is independent of η_n. The result follows after replacing $\mathcal{L}_n$ and $\mathcal{L}_n^{\text{opt}}$ and performing simple algebraic calculations.
4) Replacing $\boldsymbol{\theta}_{\text{o}}$ and dropping out the terms that contain the term ϵ_n, the result is obtained. □

Remark 3. *Observe that, while the first two results can be regarded as special cases of the distributed case (see proposition 3 and the related discussion in section III), the two last ones describe more accurately the evolution of the solution in terms of mean square stability, than the one given in proposition 4, for the general distributed scheme (see Appendix C, where no formula for the matrix $\boldsymbol{B}_n$ is given). This becomes possible because the related formulas take a much simpler form, if the graph structure is reduced to a single node.*

In order to illustrate the performance of the proposed algorithm and compare its behavior to the other variants of KLMS, we also present some related simulations. We choose the Quantized KLMS (QKLMS) [39] as a reference, since this is one of the most effective and fast KLMS pruning methods. In all experiments, that are presented in this section (described below), we use the same kernel parameter, i.e., σ, for both RFFKLMS and QKLMS as well as the same step-update parameter μ. The quantization parameter q of the QKLMS controls the size of the dictionary. If this is too large, then the dictionary will be small and the achieved MSE at steady state will be large. Typically, however, there is a value for q for which the best possible MSE (which is very close to the MSE of the unsparsified version) is attained at steady state, while any smaller quantization sizes provide negligible improvements (albeit at significantly increased complexity). In all experimental set-ups, we tuned q (using multiple trials) so that it leads to the best performance. On the other hand, the performance of RFFKLMS depends largely on D, which controls the quality of the kernel approximation. Similar to the case of QKLMS, there is a value for D so that RFFKLMS attains its lowest steady-state MSE, while larger values provide negligible improvements. For our experiments, the chosen values for q and D provide results so that to trace out the results provided by the original (unsparsified) KLMS. Table V gives the mean training times for QKLMS and RFFKLMS on the same i7-3770 machine using a MatLab implementation

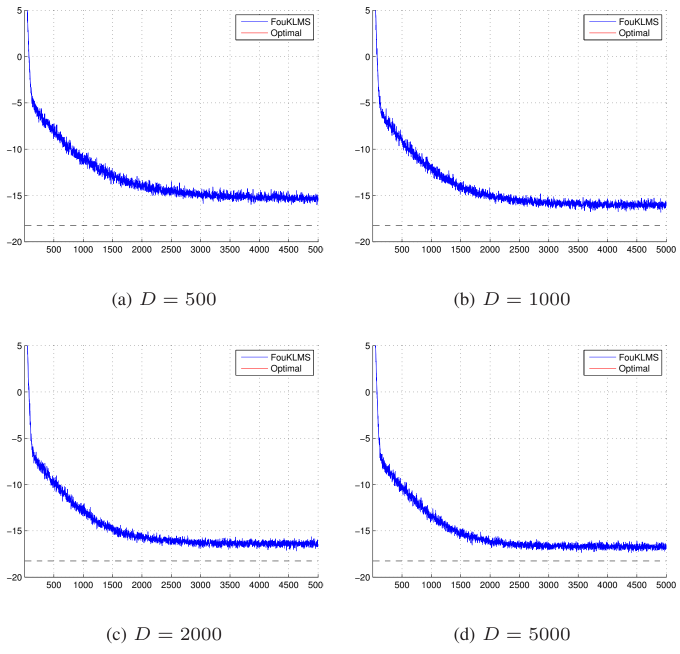


Fig. 2. Simulations of RFFKLMS (with various values of D) applied on data pairs generated by (19). The results are averaged over 500 runs. The horizontal dashed line in the figure represents the approximation of the steady-state MSE given in proposition 5.

(both algorithms were optimized for speed). We note that the complexity of the RFFKLMS is $\mathcal{O}(Dd)$, while the complexity of QKLMS is $\mathcal{O}(M_D d)$, where M_D is the size of the dictionary. Our experiments indicate that in order to obtain similar error floors, the required execution time of RFFKLMS is lower than that of QKLMS, although the latter uses a much smaller dictionary size M_D, than the size of the transformed space D. Concerning complexity, one has to take into account the fact that the QKLMS requires a sequential search of the dictionary at each step, to find the closest center.

1) Example 5. A linear expansion in terms of Kernels: Similar to example 1 in section III-C, we generate 5000 data pairs using (19) and the same parameters (for only one node). Figure 2 shows the evolution of the MSE for 500 realizations of the experiment over different values of D. The algorithm reaches steady-state around $n = 3000$. The attained MSE is getting closer to the approximation given in proposition 5 (dashed line in the figure) as D increases. Figure 3(a) compares the performances of RFFKLMS and QKLMS for this particular set-up for 500 realizations of the experiment using 8000 data pairs. The quantization size of QKLMS was set to $q = 5$ leading to an average dictionary size $M_D = 1088$ and the number of Fourier features for the RFFKLMS was set to $D = 2500$.

2) Example 6: Next, we use the same non-linear model as in example 2 of section III, i.e., $y_n = \boldsymbol{w}_0^T \boldsymbol{x}_n + 0.1 \cdot (\boldsymbol{w}_1^T \boldsymbol{x}_n)^2 + \eta_n$. The parameters of the model and the RFF-KLMS are the same as in example 1. The quantization size of the QKLMS was set to $q = 5$, leading to an average dictionary size $M_D = 100$. The number of Fourier features for the RFFKLMS was set to $D = 300$. Figure 3(b) shows the evolution of the MSE

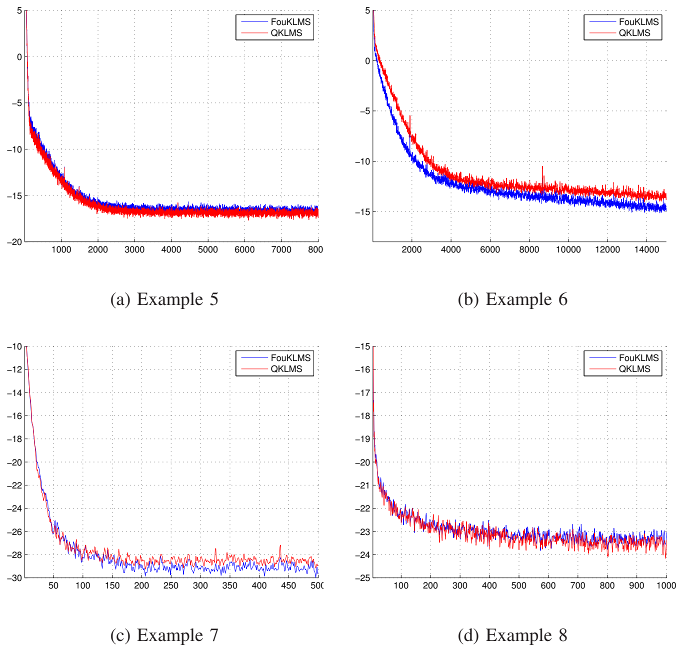


Fig. 3. Comparing the performances of RFFKLMS and the QKLMS.

TABLE V
MEAN TRAINING TIMES FOR QKLMS AND RFFKLMS.

Experiment	QKLMS time	RFFKLMS time	QKLMS dictionary size
Example 5	0.55 sec	0.35 sec	$M_D = 1088$
Example 6	0.47 sec	0.15 sec	$M_D = 104$
Example 7	0.02 sec	0.0057 sec	$M_D = 7$
Example 8	0.03 sec	0.008 sec	$M_D = 32$

for both QKLMS and RFFKLMS running 1000 realizations of the experiment over 15000 samples.

3) Example 7: Here we adopt the same chaotic series model as in example 3 of section III-C, with the same parameters. Figure 3(c) shows the evolution of the MSE for both QKLMS and RFFKLMS running 1000 realizations of the experiment over 500 samples. The quantization parameter q for the QKLMS was set to $q = 0.01$, leading to an average dictionary size $M_D = 7$. The number of Fourier features for the RFFKLMS was set to $D = 100$.

4) Example 8: For the final example, we use the chaotic series model of example 4 in section III-C with the same parameters. Figure 3(d) shows the evolution of the MSE for both QKLMS and RFFKLMS running 1000 realizations of the experiment over 1000 samples. The parameter q was set to $q = 0.01$, leading to $M_D = 32$. The number of Fourier features for the RFFKLMS was set to $D = 200$.

V. CONCLUSION

We have presented a complete fixed-budget framework for non-linear online distributed learning in the context of RKHS. The proposed scheme achieves asymptotic consensus under some reasonable assumptions. Furthermore, we showed that the respective regret bound grows sublinearly with time. In the case of a network comprising only one node, the proposed method can be regarded as a fixed budget alternative for online kernel-based learning. The presented simulations validate the theoretical results and demonstrate the effectiveness of the proposed scheme.

Appendix A
Proof of Proposition 2

In the following, we will use the notation $\mathcal{L}_{k,n}(\boldsymbol{\theta}) := \mathcal{L}(\boldsymbol{x}_{k,n}, y_{k,n}, \boldsymbol{\theta})$ to shorten the respective equations. Choose any $\boldsymbol{g} \in \mathcal{B}_{[\mathbf{0}_D, U_2]}$. It holds that

$$\begin{aligned}\|\boldsymbol{\psi}_{k,n} - \boldsymbol{g}\|^2 - \|\boldsymbol{\theta}_{k,n} - \boldsymbol{g}\|^2 &= -\|\boldsymbol{\psi}_{k,n} - \boldsymbol{\theta}_{k,n}\|^2 \\ &- 2\langle \boldsymbol{\theta}_{k,n} - \boldsymbol{\psi}_{k,n}, \boldsymbol{\psi}_{k,n} - \boldsymbol{g}\rangle = -\mu_n^2 \|\nabla \mathcal{L}_{k,n}(\boldsymbol{\psi}_{k,n})\|^2 \\ &+ 2\mu_n \langle \nabla \mathcal{L}_{k,n}(\boldsymbol{\psi}_{k,n}), \boldsymbol{\psi}_{k,n} - \boldsymbol{g}\rangle. \end{aligned} \tag{20}$$

Moreover, as $\mathcal{L}_{k,n}$ is convex, we have:

$$\mathcal{L}_{k,n}(\boldsymbol{\theta}) \geq \mathcal{L}_{k,n}(\boldsymbol{\theta}') + \langle \boldsymbol{h}, \boldsymbol{\theta} - \boldsymbol{\theta}'\rangle, \tag{21}$$

for all $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathrm{dom}(\mathcal{L}_{k,n})$ where $\boldsymbol{h} := \nabla \mathcal{L}_{k,n}(\boldsymbol{\theta})$ is the gradient (for a differentiable cost function) or a subgradient (for the case of a non–differentiable cost function). From (20), (21) and the boundness of the (sub)gradient we take

$$\begin{aligned}\|\boldsymbol{\psi}_{k,n} - \boldsymbol{g}\|^2 - \|\boldsymbol{\theta}_{k,n} - \boldsymbol{g}\|^2 \geq -\mu_n^2 U^2 \\ - 2\mu_n(\mathcal{L}_{k,n}(\boldsymbol{g}) - \mathcal{L}_{k,n}(\boldsymbol{\psi}_{k,n})),\end{aligned} \tag{22}$$

where U is an upper bound for the (sub)gradient. Recall that for the whole network we have: $\underline{\boldsymbol{\psi}}_n = \boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1}$ and that for any doubly stochastic matrix, $\boldsymbol{A}$, its norm equals to its largest eigenvalue, i.e., $\|\boldsymbol{A}\| = \lambda_{\max} = 1$. A respective eigenvector is $\underline{\boldsymbol{g}} = (\boldsymbol{g}^T \dots, \boldsymbol{g}^T)^T \in \mathbb{R}^{DK}$, hence it holds that $\underline{\boldsymbol{g}} = \boldsymbol{A}\underline{\boldsymbol{g}}$ and

$$\begin{aligned}\|\underline{\boldsymbol{\psi}}_n - \underline{\boldsymbol{g}}\| = \|\boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1} - \boldsymbol{A}\underline{\boldsymbol{g}}\| \leq \|\boldsymbol{A}\|\|\underline{\boldsymbol{\theta}}_{n-1} - \underline{\boldsymbol{g}}\| \\ = \|\underline{\boldsymbol{\theta}}_{n-1} - \underline{\boldsymbol{g}}\|\end{aligned} \tag{23}$$

where $\underline{\boldsymbol{\psi}}_n = (\boldsymbol{\psi}_n^T, \dots, \boldsymbol{\psi}_n^T)^T \in \mathbb{R}^{DK}$. Going back to (22) and summing over all $k \in \mathcal{N}$, we have:

$$\begin{aligned}\sum_{k\in\mathcal{N}} (\|\boldsymbol{\psi}_{k,n} - \boldsymbol{g}\|^2 - \|\boldsymbol{\theta}_{k,n} - \boldsymbol{g}\|^2) \geq \\ -\mu_n^2 K U^2 - 2\mu_n \sum_{k\in\mathcal{N}} (\mathcal{L}_{k,n}(\boldsymbol{g}) - \mathcal{L}_{k,n}(\boldsymbol{\psi}_{k,n})).\end{aligned} \tag{24}$$

However, for the left hand side of the inequality we obtain $\sum_{k\in\mathcal{N}}(\|\boldsymbol{\psi}_{k,n} - \boldsymbol{g}\|^2 - \|\boldsymbol{\theta}_{k,n} - \boldsymbol{g}\|^2) = \|\underline{\boldsymbol{\psi}}_n - \underline{\boldsymbol{g}}\|^2 - \|\underline{\boldsymbol{\theta}}_n - \underline{\boldsymbol{g}}\|^2$. If we combine the last relation with (23) and (24) we have

$$\begin{aligned}\|\underline{\boldsymbol{\theta}}_{n-1} - \underline{\boldsymbol{g}}\|^2 - \|\underline{\boldsymbol{\theta}}_n - \underline{\boldsymbol{g}}\|^2 \geq \\ -\mu_n^2 K U^2 - 2\mu_n \sum_{k\in\mathcal{N}} (\mathcal{L}_{k,n}(\boldsymbol{g}) - \mathcal{L}_{k,n}(\boldsymbol{\psi}_{k,n})).\end{aligned} \tag{25}$$

The last inequality leads to

$$\begin{aligned}&\frac{1}{\mu_n}\|\underline{\boldsymbol{\theta}}_{n-1} - \underline{\boldsymbol{g}}\|^2 - \frac{1}{\mu_{n+1}}\|\underline{\boldsymbol{\theta}}_n - \underline{\boldsymbol{g}}\|^2 = \\ &+ \frac{1}{\mu_n}(\|\underline{\boldsymbol{\theta}}_{n-1} - \underline{\boldsymbol{g}}\|^2 - \|\underline{\boldsymbol{\theta}}_n - \underline{\boldsymbol{g}}\|^2) \\ &+ \left(\frac{1}{\mu_n} - \frac{1}{\mu_{n+1}}\right)\|\underline{\boldsymbol{\theta}}_n - \underline{\boldsymbol{g}}\|^2 \geq \\ &- \mu_n K U^2 - 2\sum_{k\in\mathcal{N}} (\mathcal{L}_{k,n}(\underline{\boldsymbol{g}}) - \mathcal{L}_{k,n}(\boldsymbol{\psi}_{k,n})) \\ &+ 4KU_2^2\left(\frac{1}{\mu_n} - \frac{1}{\mu_{n+1}}\right),\end{aligned}$$

where we have taken into consideration, Assumption 3 and the boundeness of g. Next, summing over $i = 1, \dots, N+1$, taking into consideration that $\sum_{i=1}^N \mu_i \leq 2\mu\sqrt{N}$ (Assumption 1) and noticing that some terms telescope, we have:

$$\begin{aligned}&\frac{1}{\mu}\|\underline{\boldsymbol{\theta}}_0 - \underline{\boldsymbol{g}}\|^2 - \frac{1}{\mu_{N+1}}\|\underline{\boldsymbol{\theta}}_N - \underline{\boldsymbol{g}}\|^2 \geq -KU^2 2\mu\sqrt{N} \\ &+ 2\sum_{i=1}^N \sum_{k\in\mathcal{N}} (\mathcal{L}_{k,i}(\boldsymbol{\psi}_{k,i}) - \mathcal{L}_{k,i}(\boldsymbol{g})) + 4KU_2^2\left(\frac{1}{\mu} - \frac{\sqrt{N+1}}{\mu}\right).\end{aligned}$$

Rearranging the terms and omitting the negative ones completes the proof:

$$\begin{aligned}&\sum_{i=1}^N \sum_{k\in\mathcal{N}} (\mathcal{L}_{k,i}(\boldsymbol{\psi}_{k,i}) - \mathcal{L}_{k,i}(\boldsymbol{g})) \\ &\leq \frac{1}{2\mu}\|\underline{\boldsymbol{\theta}}_0 - \underline{\boldsymbol{g}}\|^2 + KU^2\mu\sqrt{N} + 2KU_2^2\frac{\sqrt{N+1}}{\mu} \\ &\leq \frac{1}{2\mu}\|\underline{\boldsymbol{\theta}}_0 - \underline{\boldsymbol{g}}\|^2 + KU^2\mu\sqrt{N} + 2KU_2^2\frac{\sqrt{N}+1}{\mu}.\end{aligned}$$

Appendix B
Proof of Proposition 3

For the whole network, the step update of RFF-DKLMS can be recasted as

$$\underline{\boldsymbol{\theta}}_n = \boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1} + \mu \boldsymbol{V}_n \underline{\boldsymbol{\varepsilon}}_n, \tag{26}$$

where $\underline{\boldsymbol{\varepsilon}}_n = (\varepsilon_{1,n}, \varepsilon_{2,n}, \dots, \varepsilon_{K,n})^T$ and $\varepsilon_{k,n} = y_{k,n} - \boldsymbol{\psi}_{k,n}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n})$, or equivalently, $\underline{\boldsymbol{\varepsilon}}_n = \underline{\boldsymbol{y}}_n - \boldsymbol{V}_n^T \boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1}$. If we define $\boldsymbol{U}_n = \underline{\boldsymbol{\theta}}_n - \underline{\boldsymbol{\theta}}_o$ and take into account that $\boldsymbol{A}\underline{\boldsymbol{g}} = \underline{\boldsymbol{g}}$, for all $\underline{\boldsymbol{g}} \in \mathbb{R}^{DK}$, such that $\underline{\boldsymbol{g}} = (\boldsymbol{g}^T, \boldsymbol{g}^T, \dots, \boldsymbol{g}^T)^T$ for $\boldsymbol{g} \in \mathbb{R}^D$, we obtain:

$$\begin{aligned}\boldsymbol{U}_n &= \boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1} + \mu \boldsymbol{V}_n(\underline{\boldsymbol{y}}_n - \boldsymbol{V}_n^T \boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1}) - \underline{\boldsymbol{\theta}}_o \\ &= \boldsymbol{A}(\underline{\boldsymbol{\theta}}_{n-1} - \underline{\boldsymbol{\theta}}_o) + \mu \boldsymbol{V}_n(\boldsymbol{V}_n^T \underline{\boldsymbol{\theta}}_o + \underline{\boldsymbol{\epsilon}}_n + \underline{\boldsymbol{\eta}}_n - \boldsymbol{V}_n^T \boldsymbol{A}\underline{\boldsymbol{\theta}}_{n-1}) \\ &= \boldsymbol{A}\boldsymbol{U}_{n-1} - \mu \boldsymbol{V}_n \boldsymbol{V}_n^T \boldsymbol{A}\boldsymbol{U}_{n-1} + \mu \boldsymbol{V}_n \underline{\boldsymbol{\epsilon}}_n + \mu \boldsymbol{V}_n \underline{\boldsymbol{\eta}}_n\end{aligned}$$

If we take the mean values and assume that $\boldsymbol{\theta}_{k,n}$ and $\boldsymbol{z}_\Omega(\boldsymbol{x}_{k,n})$ are independent for all $k = 1, \dots, K$, $n = 1, 2, \dots$, we have

$$E[\boldsymbol{U}_n] = (I_{KD} - \mu \boldsymbol{R})\boldsymbol{A}E[\boldsymbol{U}_{n-1}] + \mu E[\boldsymbol{V}_n \underline{\boldsymbol{\epsilon}}_n] + \mu E[\boldsymbol{V}_n \underline{\boldsymbol{\eta}}_n].$$

Taking into account that $\boldsymbol{\eta}_n$ and $\boldsymbol{V}_n$ are independent, that $E[\boldsymbol{\eta}_n] = \mathbf{0}$ and that for large enough D we have $E[\boldsymbol{V}_n \boldsymbol{\epsilon}_n] \approx \mathbf{0}$, we can take $E[\boldsymbol{U}_n] \approx ((I_{KD} - \mu \boldsymbol{R})\boldsymbol{A})^{n-1} E[\boldsymbol{U}_1]$. Hence, if all the eigenvalues of $(I_{KD} - \mu \boldsymbol{R})\boldsymbol{A}$ have absolute value less than 1, we have that $E[\boldsymbol{U}_n] \to \mathbf{0}$. However, since $\boldsymbol{A}$ is a doubly stochastic matrix we have $\|\boldsymbol{A}\| \leq 1$ and

$$\|(I_{KD} - \mu \boldsymbol{R})\boldsymbol{A}\| \leq \|I_{KD} - \mu \boldsymbol{R}\|\|\boldsymbol{A}\| \leq \|I_{KD} - \mu \boldsymbol{R}\|.$$

Moreover, as $I_{KD} - \mu \boldsymbol{R}$ is a diagonal block matrix, its eigenvalues are identical to the eigenvalues of its blocks, i.e., the eigenvalues of $I_D - \mu R_{zz}$. Hence, a sufficient condition for convergence is $|1 - \mu \lambda_D(R_{zz})| < 1$, which gives the result. □

Remark 4. *Observe that* $|\lambda_{\max}((I_{KD} - \mu \boldsymbol{R})\boldsymbol{A})| \leq |\lambda_{\max}((I_{KD} - \mu \boldsymbol{R})I_{KD})|$*, which means that the spectral* *radius of $(I_{KD} - \mu\boldsymbol{R})\boldsymbol{A}$ is generally smaller than that of $(I_{KD} - \mu\boldsymbol{R})I_{KD}$ (which corresponds to the non-cooperative protocol). Hence, cooperation under the diffusion rationale has a stabilizing effect on the network [8].*

Appendix C
Proof of Proposition 4

Let $\boldsymbol{B}_n = E[\boldsymbol{U}_n\boldsymbol{U}_n^T]$, where $\boldsymbol{U}_n = \boldsymbol{A}\boldsymbol{U}_{n-1} - \mu\boldsymbol{V}_n\boldsymbol{V}_n^T\boldsymbol{A}\boldsymbol{U}_{n-1} + \mu\boldsymbol{V}_n\underline{\boldsymbol{\epsilon}}_n + \mu\boldsymbol{V}_n\underline{\boldsymbol{\eta}}_n$. Taking into account that the noise is i.i.d., independent from $\boldsymbol{U}_n$ and $\boldsymbol{V}_n$ and that $\boldsymbol{\epsilon}_n$ is close to zero (if D is sufficiently large), we can take that:

$$\begin{aligned}\boldsymbol{B}_n =& \boldsymbol{A}\boldsymbol{B}_{n-1}\boldsymbol{A}^T - \mu\boldsymbol{A}\boldsymbol{B}_{n-1}\boldsymbol{A}^T\boldsymbol{R} - \mu\boldsymbol{R}\boldsymbol{A}\boldsymbol{B}_{n-1}\boldsymbol{A}^T\\ &+ \mu^2\sigma_\eta^2\boldsymbol{R} + \mu^2E[\boldsymbol{V}_n\boldsymbol{V}_n^T\boldsymbol{A}\boldsymbol{U}_{n-1}\boldsymbol{U}_{n-1}^T\boldsymbol{A}^T\boldsymbol{V}_n\boldsymbol{V}_n^T].\end{aligned}$$

For sufficiently small step-sizes, the rightmost term can be neglected [55], [51], hence we can take the simplified form

$$\begin{aligned}\boldsymbol{B}_n =& \boldsymbol{A}\boldsymbol{B}_{n-1}\boldsymbol{A}^T - \mu\boldsymbol{A}\boldsymbol{B}_{n-1}\boldsymbol{A}^T\boldsymbol{R} - \mu\boldsymbol{R}\boldsymbol{A}\boldsymbol{B}_{n-1}\boldsymbol{A}^T\\ &+ \mu^2\sigma_\eta^2\boldsymbol{R}. \end{aligned}\tag{27}$$

Next, we observe that $\boldsymbol{B}_n$, $\boldsymbol{R}$ and $\boldsymbol{A}$ can be regarded as block matrices, that consist of $K \times K$ blocks with size $D \times D$. We will vectorize equation (27) using the $\mathrm{vecb_r}$ operator, as this has been defined in [56]. Assuming a block-matrix C:

$$C = \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1K}\\ C_{21} & C_{22} & \cdots & C_{2K}\\ \vdots & \vdots & & \vdots\\ C_{K1} & C_{K2} & \cdots & C_{KK}\end{pmatrix},$$

the $\mathrm{vecb_r}$ operator applies the following vectorization:

$$\begin{aligned}\mathrm{vecb_r}\, C = (&\mathrm{vec}\, C_{11}^T, \mathrm{vec}\, C_{12}^T, \dots, \mathrm{vec}\, C_{1K}^T, \dots,\\ &\mathrm{vec}\, C_{K1}^T\, \mathrm{vec}\, C_{K2}^T, \dots, \mathrm{vec}\, C_{KK}^T)^T.\end{aligned}$$

Moreover, it is closely related to the following (unbalanced) block Kronecker product:

$$D \boxtimes C = \begin{pmatrix} D\otimes C_{11} & D\otimes C_{12} & \cdots & D\otimes C_{1K}\\ D\otimes C_{21} & D\otimes C_{22} & \cdots & D\otimes C_{2K}\\ \vdots & \vdots & & \vdots\\ D\otimes C_{K1} & D\otimes C_{K2} & \cdots & D\otimes C_{KK}\end{pmatrix}.$$

The interested reader can delve into the details of the $\mathrm{vecb_r}$ operator and the unbalanced block Kronecker product in [56]. Here, we limit our interest to the following properties:

1) $\mathrm{vecb_r}(DCE^T) = (E \boxtimes D)\,\mathrm{vecb_r}\, C$.
2) $(C \boxtimes D)(E \boxtimes F) = CE \boxtimes DF$.

Thus, applying the $\mathrm{vecb_r}$ operator, on both sizes of (27) we take $\boldsymbol{b}_n = (\boldsymbol{A} \boxtimes \boldsymbol{A})\boldsymbol{b}_{n-1} - \mu\left((\boldsymbol{R}\boldsymbol{A}) \boxtimes \boldsymbol{A}\right)\boldsymbol{b}_{n-1} - \mu\left((\boldsymbol{A}) \boxtimes \boldsymbol{R}\boldsymbol{A}\right)\boldsymbol{b}_{n-1} + \mu^2\sigma_\eta^2\boldsymbol{r}$, where $\boldsymbol{b}_n = \mathrm{vecb_r}\, \boldsymbol{B}_n$ and $\boldsymbol{r} = \mathrm{vecb_r}\, \boldsymbol{R}$. Exploiting the second property, we can take:

$$\begin{aligned}(\boldsymbol{R}\boldsymbol{A}) \boxtimes \boldsymbol{A} &= (\boldsymbol{R}\boldsymbol{A}) \boxtimes (\boldsymbol{I}_{DK}\boldsymbol{A}) = (\boldsymbol{R} \boxtimes \boldsymbol{I}_{DK})(\boldsymbol{A} \boxtimes \boldsymbol{A}),\\ \boldsymbol{A} \boxtimes (\boldsymbol{R}\boldsymbol{A}) &= (\boldsymbol{I}_{DK}\boldsymbol{A}) \boxtimes (\boldsymbol{R}\boldsymbol{A}) = (\boldsymbol{I}_{DK} \boxtimes \boldsymbol{R})(\boldsymbol{A} \boxtimes \boldsymbol{A}).\end{aligned}$$

Hence, we finally get:

$$\begin{aligned}\boldsymbol{b}_n =& \left(\boldsymbol{I}_{D^2K^2} - \mu\left(\boldsymbol{R} \boxtimes \boldsymbol{I}_{DK} - \boldsymbol{I}_{DK} \boxtimes \boldsymbol{A}\right)\right)(\boldsymbol{A} \boxtimes \boldsymbol{A})\boldsymbol{b}_{n-1}\\ &+ \mu^2\sigma_\eta^2\boldsymbol{r},\end{aligned}$$

which gives the result.

References

[1] K. Slavakis, G. Giannakis, and G. 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] C. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," *Advances in neural information processing systems*, vol. 19, p. 281, 2007.

[3] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology*. ACM, 2011, pp. 530–533.

[4] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.

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

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

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

[8] C. G. Lopes and A. 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, July 2008.

[9] R. L. Cavalcante, I. Yamada, and B. Mulgrew, "An adaptive projected subgradient approach to learning in diffusion networks," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2762–2774, 2009.

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

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

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical Methods*, 2nd ed. Athena-Scientific, 1999.

[13] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[14] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4129–4144, 2014.

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

[16] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[17] Z. J. Towfic, J. Chen, and A. H. Sayed, "On distributed online classification in the midst of concept drifts," *Neurocomputing*, vol. 112, pp. 138–152, 2013.

[18] B. Scholkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

[19] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge UK: Cambridge University Press, 2004.

[20] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4^{th} ed. Academic Press, 2009.

[21] R. Mitra and V. Bhatia, "The diffusion-KLMS algorithm," in *ICIT, 2014*, Dec 2014, pp. 256–259.

[22] W. Gao, J. Chen, C. Richard, and J. Huang, "Diffusion adaptation over networks with kernel least-mean-square," in *CAMSAP*, 2015.

[23] S. Chouvardas and M. Draief, "A diffusion kernel LMS algorithm for nonlinear adaptive networks," in *ICASSP*, 2016.

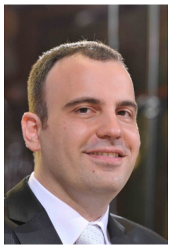
[24] A. Rahimi and B. Recht, "Random features for large scale kernel machines," in *NIPS*, vol. 20, 2007.

[25] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[26] G. Wahba, *Spline Models for Observational Data, volume 59 of CBMS-NSF Regional Conference Series in Applied Mathematics*. Philadelphia: SIAM, 1990.

[27] W. Liu, P. Pokharel, and J. C. Principe, "The kernel Least-Mean-Square algorithm," *IEEE Transanctions on Signal Processing*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[28] P. Bouboulis and S. 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.
[29] S. Van Vaerenbergh, J. Via, and I. Santamana, "A sliding-window kernel rls algorithm and its application to nonlinear channel identification," in *ICASSP*, vol. 5, may 2006, p. V.
[30] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transanctions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.
[31] K. Slavakis and S. Theodoridis, "Sliding window generalized kernel affine projection algorithm using projection mappings," *EURASIP Journal on Advances in Signal Processing*, vol. 19, p. 183, 2008.
[32] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Adaptive multiregression in reproducing kernel Hilbert spaces: the multiaccess MIMO channel case," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23(2), pp. 260–276, 2012.
[33] K. Slavakis, S. Theodoridis, and I. Yamada, "On line kernel-based classification using adaptive projection algorithms," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 2781–2796, Jul. 2008.
[34] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for svm," *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10107-010-0420-4
[35] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Online learning in reproducing kernel Hilbert spaces," in *Signal Processing Theory and Machine Learning*, ser. Academic Press Library in Signal Processing, R. Chellappa and S. Theodoridis, Eds. Academic Press, 2014, pp. 883–987.
[36] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering*. Hoboken, NJ: Wiley, 2010.
[37] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058 –1067, march 2009.
[38] W. Gao, J. Chen, C. Richard, and J. Huang, "Online dictionary learning for kernel LMS," *IEEE Transactions on Signal Processing*, vol. 62, no. 11, pp. 2765 – 2777, 2014.
[39] B. Chen, S. Zhao, P. Zhu, and J. Principe, "Quantized kernel least mean square algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 22 –32, jan. 2012.
[40] S. Zhao, B. Chen, C. Zheng, P. Zhu, and J. Principe, "Self-organizing kernel adaptive filtering," *EURASIP Journal on Advances in Signal Processing*, (to appear).
[41] C. Williams and M. Seeger, "Using the Nystrom method to speed up kernel machines," in *NIPS*, vol. 14, 2001, pp. 682 – 688.
[42] P. Drineas and M. W. Mahoney, "On the Nystrom method for approximating a gram matrix for improved kernel-based learning," *JMLR*, vol. 6, pp. 2153 – 2175, 2005.
[43] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: replacing minimization with randomization in learning," in *NIPS*, vol. 22, 2009, pp. 1313 – 1320.
[44] D. J. Sutherland and J. Schneider, "On the error of random Fourier features," in *UAI*, 2015.
[45] T. Yang, Y.-F. Li, M. Mahdavi, J. Rong, and Z.-H. Zhou, "Nyström method vs random Fourier features: A theoretical and empirical comparison," in *NIPS*, vol. 25, 2012, pp. 476–484.
[46] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections: a unifying framework for linear and nonlinear classification and regression tasks," *Signal Procesing Magazine*, vol. 28, no. 1, pp. 97–123, Jan. 2011.
[47] J. Kivinen, A. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Transanctions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.
[48] L. Bottou, "http://leon.bottou.org/projects/lasvm."
[49] MIT, Strategic Engineering Research Group, MatLab Tools for Network analysis, "http://strategic.mit.edu/."
[50] C. G. Lopes and A. 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.
[51] F. S. Cattiveli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, 2010.
[52] W. Parreira, J. Bermudez, C. Richard, and J.-Y. Tourneret, "Stochastic behavior analysis of the gaussian kernel least-mean-square algorithm," *Signal Processing, IEEE Transactions on*, vol. 60, no. 5, pp. 2208–2222, May 2012.
[53] A. Singh, N. Ahuja, and P. Moulin, "Online learning with kernels: Overcoming the growing sum problem," *MLSP*, September 2012.
[54] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: A random Fourier feature perspective," *SSP*, 2016.
[55] S. C. Douglas and M. Rupp, "Digital signal processing fundamentals," V. K. Madisetti, Ed. CRC Press, 2009, ch. 19. Convergence Issues in the LMS Adaptive Filter, pp. 1–21.
[56] R. H. Koning and H. Neudecker, "Block Kronecker products and the vecb operator," *Linear Algebra and its Applications*, vol. 149, pp. 165–184, 1991.



Pantelis Bouboulis Pantelis Bouboulis received the B.Sc. degree in Mathematics and the M.Sc. and Ph.D. degrees in Informatics and Telecommunications from the National and Kapodistrian University of Athens, Greece, in 1999, 2002 and 2006, respectively. From 2007 till 2008, he served as an Assistant Professor in the Department of Informatics and Telecommunications, University of Athens. In 2010, he has received the Best scientific paper award for a work presented in the International Conference on Pattern Recognition, Istanbul, Turkey. Currently, he is a Research Fellow at the Signal and Image Processing laboratory of the department of Informatics and Telecommunications of the University of Athens and he teaches mathematics at the Zanneio Model Experimental Lyceum of Pireas. From 2012 since 2014, he served as an Associate Editor of the IEEE Transactions of Neural Networks and Learning Systems. His current research interests lie in the areas of machine learning, fractals, signal and image processing.

Symeon Chouvardas Symeon Chouvardas received the B.Sc., M.Sc. (honors) and Ph.D. degrees from National and Kapodistrian University of Athens, Greece, in 2008, 2011, and 2013, respectively. He was granted a Heracletus II Scholarship from GSRT (Greek Secretariat for Research and Technology) to pursue his PhD. In 2010 he was awarded with the Best Student Paper Award for the International Workshop on Cognitive Information Processing (CIP), Elba, Italy and in 2016 the Best Paper Award for the International Conference on Communications, ICC, Kuala Lumpur, Malaysia. His research interests include: machine learning, signal processing, compressed sensing and online learning.

Sergios Theodoridis (F' 08) is currently Professor of Signal Processing and Machine Learning in the Department of Informatics and Telecommunications of the University of Athens. His research interests lie in the areas of Adaptive Algorithms, Distributed and Sparsity-Aware Learning, Machine Learning and Pattern Recognition, Signal Processing for Audio Processing and Retrieval. He is the author of the book Machine Learning: A Bayesian and Optimization Perspective, Academic Press, 2015, the co-author of the best-selling book Pattern Recognition, Academic Press, 4th ed. 2009, the co-author of the book Introduction to Pattern Recognition: A MATLAB Approach, Academic Press, 2010, the co-editor of the bookEfficient Algorithms for Signal Processing and System Identification, Prentice Hall 1993, and the co-author of three books in Greek, two of them for the Greek Open University. He currently serves as Editor-in-Chief for the IEEE Transactions on Signal Processing. He is Editor-in-Chief for the Signal Processing Book Series, Academic Press and co-Editor in Chief for the E-Reference Signal Processing, Elsevier. He is the co-author of seven papers that have received Best Paper Awards including the 2014 IEEE Signal Processing Magazine best paper award and the 2009 IEEE Computational Intelligence Society Transactions on Neural Networks Outstanding Paper Award. He is the recipient of the 2014 IEEE Signal Processing Society Education Award and the 2014 EURASIP Meritorious Service Award. He has served as a Distinguished Lecturer for the IEEE SP and CAS Societies. He was Otto Monstead Guest Professor, Technical University of Denmark, 2012, and holder of the Excellence Chair, Dept. of Signal Processing and Communications, University Carlos III, Madrid, Spain, 2011. He has served as President of the European Association for Signal Processing (EURASIP), as a member of the Board of Governors for the IEEE CAS Society, as a member of the Board of Governors (Member-at-Large) of the IEEE SP Society and as a Chair of the Signal Processing Theory and Methods (SPTM) technical committee of IEEE SPS. He is Fellow of IET, a Corresponding Fellow of the Royal Society of Edinburgh (RSE), a Fellow of EURASIP and a Fellow of IEEE.