EFFICIENT KLMS AND KRLS ALGORITHMS: A RANDOM FOURIER FEATURE PERSPECTIVE

Pantelis Bouboulis, Spyridon Pougkakiotis, S. Theodoridis

University of Athens
Department of Informatics and Telecomunications
Athens, Greece.
panbouboulis@gmail.com, sdi1200151@di.uoa.gr, stheodor@di.uoa.gr

ABSTRACT

We present a new framework for online Least Squares algorithms for nonlinear modeling in RKH spaces (RKHS). Instead of implicitly mapping the data to a RKHS (e.g., kernel trick), we map the data to a finite dimensional Euclidean space, using random features of the kernel's Fourier transform. The advantage is that, the inner product of the mapped data approximates the kernel function. The resulting "linear" algorithm does not require any form of sparsification, since, in contrast to all existing algorithms, the solution's size remains fixed and does not increase with the iteration steps. As a result, the obtained algorithms are computationally significantly more efficient compared to previously derived variants, while, at the same time, they converge at similar speeds and to similar error floors.

Index Terms— KLMS, Kernel Adaptive filter, Random Fourier Features, Kernel Least Mean Squares, Kernel LMS, Kernel RLS

1. INTRODUCTION

Online learning in RKH spaces has attracted a lot of interest over the last years, see, e.g., [1, 2, 3, 4, 5, 6, 7, 8]. The Kernel Least Mean Square (KLMS) algorithm, introduced in [9, 10], presents a simple and efficient method to address non linear adaptive filtering tasks. Considering a sequentially arriving data of the form $\{(\boldsymbol{x}_n, y_n),\ n = 1, 2, \dots\}$, where $\boldsymbol{x}_n \in \mathbb{R}^d$, $y_n \in \mathbb{R}$, generated by a non-linear model, KLMS's mechanism can be summarized as follows: (a) map each arriving input datum, $\boldsymbol{x}_n$, to an infinite dimensional Hilbert space $\mathcal{H}$, using a specific kernel κ and (b) apply the LMS rationale to the transformed data, i.e., $\{(\kappa(\boldsymbol{x}_n, \cdot), y_n),\ n = 1, 2, \dots\}$. Its main drawback is that the solution is given in terms of a linear expansion of kernel functions (centered at the input data points $\boldsymbol{x}_n$), which grows infinitely large (proportionally to n), rendering its application prohibitive both in terms of memory and computational resources. The centers, $\boldsymbol{x}_n$, that make up the linear expansion of the solution, are said to constitute the *dictionary*. In practice, sparsification methods are applied to keep the size of the dictionary sufficiently small and make the algorithm computationally tractable. These methods adopt a suitably selected criterion to decide whether a particular datum (i.e., $\boldsymbol{x}_n$) will be included in the dictionary or not. Popular variations include the quantization [11], the novelty [9], the coherence [12] and the surprise [13] criteria.

Although the aforementioned sparsification techniques are able to reduce the size of the expansion significantly, they, too, require significant computational resources, even when the dictionary is small. This is due to the fact that at each iteration step, n, a sequential search over all the current dictionary elements has to be performed, in order to determine whether the new center, $\boldsymbol{x}_n$, will be added to the dictionary or not. Another important issue is the dimension of the input space. If this is small (e.g., $d < 5$), then the aforementioned sparsification strategies may result in dictionaries with a few dozens elements, without compromising Mean Square Error (MSE) performance. However, if this dimension grows larger, then these methods will inevitably give dictionaries with several thousands elements or more rendering KLMS prohibitively demanding due to the sequential search over large dictionaries. Furthermore, from a theoretical point of view, such approaches are not elegant, in the sense that they build around "ad hoc" arguments, which, also, complicate the corresponding theoretical analysis.

The aforementioned difficulties have limited the extension of KLMS to more general settings, such as in distributed learning. In this case, the exchange of dictionaries among the network's nodes increase the network's load significantly [14, 15, 16]. More importantly, as each node should match its dictionary with the dictionaries of its neighbors (applying multiple sequential searches) the required computational resources become quite demanding. In the present work, we follow a different rationale. Instead of mapping the input data to an infinite dimensional Reproducing Kernel Hilbert Space, induced by the selected kernel, and subsequently sparsifying the solution, we map the input data to a finite (although larger than the input one) dimensional Euclidian space $\mathbb{R}^D$. However, this mapping is done in a sensible way that cares for a good *approximation* of the kernel evaluations. The mapping to $\mathbb{R}^D$ is carried out using random features of the kernel's Fourier transform [17, 18, 19]. Following this approach, the resulting algorithm, which we call *Random Fourier Features Kernel LMS* or RFFKLMS for short, leads naturally to a standard linear LMS, with a *fixed-size* solution (i.e., a vector in $\mathbb{R}^D$); thus, no special sparsification techniques are needed. RFFKLMS is computationally lighter than various variants of KLMS, while at the same time it exhibits the same MSE performance (for sufficient large D). Similar arguments as before hold true for the case of the KRLS.

Section 2 briefly describes the rationale behind the standard KLMS with the quantization sparsification strategy. Sections 3 and 4 present the theory of approximating shift-invariant kernels with random features of their Fourier Transform and the new linearized implementation of the KLMS using this approximation. Simulations are given in section 5. Section 6 briefly describes the "linearized" version of KRLS based on the random Fourier features approximation framework, while section 7 concludes the paper. In the following, matrices appear with capital letters and vectors with small bold letters.

This research was funded by the European Union (European Social Fund - ESF) through the EC - FP7 FET program HANDiCAMS.

2. THE QUANTIZED KLMS

Consider the sequence $\mathcal{D} = \{(\boldsymbol{x}_n, y_n),\ n = 1, 2, \dots\}$, where $\boldsymbol{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$. The goal of the KLMS is to learn a non-linear input-output map f, so that to minimize the MSE, i.e., $\mathcal{L}(f) = E[(y_n - f(\boldsymbol{x}_n))^2]$. Typically, we assume that f lies in a RKHS induced by the Gaussian kernel, i.e., $\kappa_\sigma(\boldsymbol{u}, \boldsymbol{v}) = e^{-\|\boldsymbol{u}-\boldsymbol{v}\|_2^2/(2\sigma^2)}$, for some $\sigma > 0$. Computing the gradient of $\mathcal{L}$ and estimating it by its current measurement (as it is typically the case in LMS), we take the solution at the next iteration, i.e., $f_n = f_{n-1} + \mu e_n \kappa(\boldsymbol{x}_n, \cdot)$, where $e_n = y_n - f_n(\boldsymbol{x}_n)$ and μ is the step-size (see [4, 13] for more). Assuming that the initial solution is zero, the solution after n steps becomes $f = \sum_{i=1}^{n} \theta_i \kappa_\sigma(\boldsymbol{x}_i, \cdot)$. As mentioned in the introduction, this linear expansion grows indefinitely as n increases; hence a sparsification strategy has to be adopted to keep the expansion's size low. In this paper, we will employ a very simple and effective strategy, which is based on the quantization of the input space [11]. At each iteration. the algorithm determines whether the new point, $\boldsymbol{x}_n$, is to be included to the list of the M expansion centers, i.e., the *dictionary* C, or not, based on its distance from C. If this distance is larger than a user-defined parameter δ (the *quantization size*), then $\boldsymbol{x}_n$ is inserted to C, otherwise the coefficient of the center that is closest to $\boldsymbol{x}_n$ is updated. The resulting algorithm is called QKLMS:

- Set $f = 0, C = \emptyset$, $M = 0$. Select the step-size μ, the parameter of the kernel σ and the quantization size ϵ.
- for $n = 1, 2, \dots$ do:
  1. Compute system's output: $\hat{y}_n = f(\boldsymbol{x}_n)$.
  2. Compute the error: $e_n = y_n - \hat{y}_n$.
  3. Compute $d_k = \|\boldsymbol{x}_n - \boldsymbol{c}_k\|^2, k = 1, \dots M$.
  4. Find $d_{\min} = \min\{d_k, k = 1, \dots M\}$ and $k_{\min} = \text{argmin}\{d_k, k = 1, \dots M\}$.
  5. If $d_{\min} < \epsilon$ then $\theta_{k_{\min}} = \theta_{k_{\min}} + \mu e_n$.
  6. else $C = C \bigcup \{\boldsymbol{x}_n\}$, $M = M + 1$, $\theta_M = \mu e_n$.

Note that, there are other sparsification strategies that can be applied, as it has been mentioned in the introduction. The difference is in the different criteria used to include (or not) a specific center into the dictionary. The QKLMS is among the most effective strategies and in the following it will be used as a representative of these methods. Results with other sparsification methods follow similar trends.

3. APPROXIMATING THE KERNEL WITH RANDOM FOURIER FEATURES

The standard implementations of KLMS can be viewed as a two step procedure. Firstly, the input data, $\boldsymbol{x}_n$, are mapped to an infinite dimensional RKHS, $\mathcal{H}$, using an implicit map $\Phi(\boldsymbol{x}_n) = \kappa(\boldsymbol{x}_n, \cdot)$, and then the standard LMS rationale is applied to the transformed data pairs, i.e. $(\Phi(\boldsymbol{x}_n), y_n)$, taking into account the so called *kernel trick*, i.e., $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \langle \Phi(\boldsymbol{x}_n), \Phi(\boldsymbol{x}_m) \rangle_{\mathcal{H}}$, to evaluate the respective inner products. However, as it has been discussed in Section 2, this leads to a solution that is expressed in terms of kernel functions, whose number keeps growing. Instead of relying on the implicit lifting provided by the kernel trick, Rahimi and Recht in [17] proposed to map the input data to a low-dimensional Euclidean space using a randomized feature map $\boldsymbol{z} : \mathbb{R}^d \to \mathbb{R}^D$, so that the kernel evaluations can be approximated as $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) \approx \boldsymbol{z}(\boldsymbol{x}_n)^T \boldsymbol{z}(\boldsymbol{x}_m)$.

As $\boldsymbol{z}$ is a finite dimensional lifting, direct fast linear methods can be applied to the transformed data (unlike the kernel's lifting Φ, which requires special treatment). Hence, if one models the system's output as $\hat{y}_n = \boldsymbol{\theta}^T \boldsymbol{z}(\boldsymbol{x}_n)$, the standard linear LMS rationale can be applied directly to estimate the solution $\boldsymbol{\theta} \in \mathbb{R}^D$ at each iteration. 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*

$$\kappa(\boldsymbol{x} - \boldsymbol{y}) = E_{\boldsymbol{\omega},b}[z_{\boldsymbol{\omega},b}(\boldsymbol{x}) z_{\boldsymbol{\omega},b}(\boldsymbol{y})], \tag{1}$$

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 (a similar rationale as the one used in Monte Carlo Methods; for Gaussian kernels such sampling is trivial):

$$\kappa(\boldsymbol{x}_n - \boldsymbol{x}_m) \approx \frac{1}{D} \sum_{i=1}^{D} z_{\boldsymbol{\omega}_i,b_i}(\boldsymbol{u}) z_{\boldsymbol{\omega}_i,b_i}(\boldsymbol{v}). \tag{2}$$

Evidently, the larger D is (up to a certain point), the better this approximation becomes. Details on the quality of this approximation can be found in [17].

4. THE RANDOM FOURIER FEATURES KERNEL LMS

In this Section, we briefly describe the proposed *linearized* KLMS, which is based on the aforementioned Fourier approximation. The main results (regarding convergence and other related properties) are given without proofs due to lack of space. Our starting point is to recast (2) in terms of Euclidean inner products. To that end, we define the map $\boldsymbol{z}_\Omega : \mathbb{R}^d \to \mathbb{R}^D$ as follows:

$$\boldsymbol{z}_\Omega(\boldsymbol{u}) = \sqrt{\frac{2}{D}} \begin{pmatrix} \cos(\boldsymbol{\omega}_1^T \boldsymbol{u} + b_1) \\ \vdots \\ \cos(\boldsymbol{\omega}_D^T \boldsymbol{u} + b_D) \end{pmatrix}, \tag{3}$$

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

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

provided that $\boldsymbol{\omega}$'s and b's are drawn as mentioned above. Hence, the kernel function can be approximated as

$$\kappa(\boldsymbol{x}_n - \boldsymbol{x}_m) \approx \boldsymbol{z}_\Omega(\boldsymbol{x}_n)^T \boldsymbol{z}_\Omega(\boldsymbol{x}_m). \tag{4}$$

Following this rationale, we propose a new variant of the KLMS, the RFFKLMS, which is actually a simple LMS on the transformed data, i.e. $\{(\boldsymbol{z}_\Omega(\boldsymbol{x}_n), y_n), n = 1, 2, \dots\}$. We model the input-output relationship as $\hat{y}_n = \boldsymbol{\theta}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_n)$, for each $\boldsymbol{x}_n$ and our goal is to evaluate $\boldsymbol{\theta} \in \mathbb{R}^D$ by minimizing the MSE, i.e., $J_n = E[e_n^2]$, at each time instant n. For the Gaussian kernel, which is employed throughout the paper, the respective Fourier transform is

$$p(\boldsymbol{\omega}) = \left(\sigma/\sqrt{2\pi}\right)^D e^{-\frac{\sigma^2 \|\boldsymbol{\omega}\|^2}{2}}, \tag{5}$$

which is actually the multivariate Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\frac{1}{\sigma^2} \boldsymbol{I}_D$. The proposed algorithm is given next:

- Set $\boldsymbol{\theta} = \mathbf{0}$. Select the step-update μ, the dimension of the new space, D and the parameter of the kernel (σ).
- Draw D samples from $p(\boldsymbol{\omega})$ and D numbers uniformly in $[0, 2\pi]$.
- for $n = 1, 2, \ldots$ do:
  1. Compute system's output: $\hat{y}_n = \boldsymbol{\theta}^T \boldsymbol{z}_\Omega(\boldsymbol{x}_n)$.
  2. Compute the error: $e_n = y_n - \hat{y}_n$.
  3. $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \mu e_n \boldsymbol{z}_\Omega(\boldsymbol{x}_n)$.

It is a matter of elementary algebra to conclude that after $n-1$ steps, the algorithm will give the following solution: $\boldsymbol{\theta} = \mu \sum_{k=1}^{n-1} e_k \boldsymbol{z}_\Omega(\boldsymbol{x}_k)$, which leads us to conclude that RFFKLMS will produce approximately the same system's output with the standard KLMS (provided that D is sufficiently large), since

$$\hat{y}_n = \mu \sum_{k=1}^{n-1} e_k \boldsymbol{z}_\Omega(\boldsymbol{x}_k)^T \boldsymbol{z}_\Omega(\boldsymbol{x}_n) \approx \mu \sum_{k=1}^{n-1} e_k \kappa_\sigma(\boldsymbol{x}_k, \boldsymbol{x}_n). \quad (6)$$

However, the major difference is that RFFKLMS provides a single vector $\boldsymbol{\theta}$ of fixed dimensions, instead of a growing expansion of kernel functions.

To study the convergence properties of RFFKLMS, we will assume henceforth that the data pairs are generated by

$$y_n = \sum_{m=1}^{M} a_m \kappa(\boldsymbol{c}_m, \boldsymbol{x}_n) + \eta_n, \quad (7)$$

where $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M$ 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 $\mathcal{N}(0, \sigma_\eta^2)$. In this setting it is not difficult to prove that the optimal solution is given by

$$\boldsymbol{\theta}_{\text{opt}} = \text{argmin} E[e_n^2] = Z_C \cdot \boldsymbol{a} + R_{zz}^{-1} E[\eta_n' \cdot \boldsymbol{z}_\Omega(\boldsymbol{x}_n)], \quad (8)$$

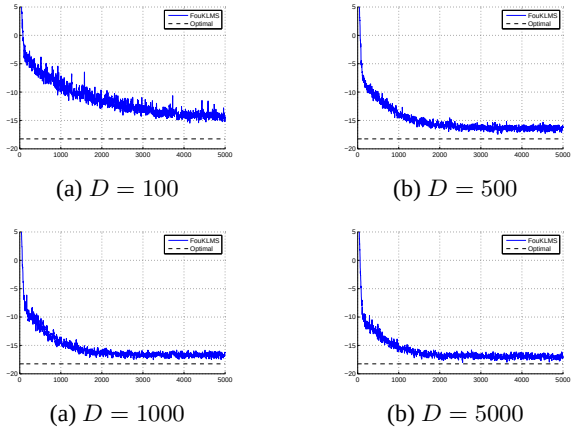
where $Z_C = (\boldsymbol{z}_\Omega(\boldsymbol{c}_1), \ldots, \boldsymbol{z}_\Omega(\boldsymbol{c}_M))^T$, $\boldsymbol{a} = (a_1, \ldots, a_M)^T$, $R_{zz} = E[\boldsymbol{z}_\Omega(\boldsymbol{x}_n) \boldsymbol{z}_\Omega(\boldsymbol{x}_n)^T]$ and η_n' is the approximation error between the noise-free component of y_n (evaluated only by the linear kernel expansion of (7)) and the approximation of this component using random Fourier features, i.e., $\eta_n = \sum_{m=1}^{M} a_m \kappa(\boldsymbol{c}_m, \boldsymbol{x}_n) - \sum_{m=1}^{M} \boldsymbol{z}_\Omega(\boldsymbol{c}_m)^T \boldsymbol{z}_\Omega(\boldsymbol{x}_n)$. Note that this error can be made very small for sufficiently large D [17]; thus, it can be eventually dropped out. Furthermore, sufficient conditions so that R_{zz} is a strictly positive definite matrix (hence invertible) have been obtained. These are summarized by:

Lemma 1. *Consider a selection of samples $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_D$, drawn from* (5) *such that $\boldsymbol{\omega}_i \neq \boldsymbol{\omega}_j$, for any $i \neq j$. Then, the matrix $R_{zz} = E[\boldsymbol{z}_\Omega(\boldsymbol{x}_n) \boldsymbol{z}_\Omega(\boldsymbol{x}_n)^T]$ is strictly positive definite.*

It is also possible, for $\boldsymbol{x}_n \sim \mathcal{N}(\mathbf{0}, \sigma_X \boldsymbol{I}_d)$, to explicitly evaluate the entries of R_{zz}:

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

As expected, the eigenvalues of R_{zz} play a pivotal role in the convergence's study of the algorithm. In the case where R_{zz} is a strictly positive definite matrix, its eigenvalues satisfy $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_D$. Applying similar assumptions as in the case of the standard LMS, we can prove the following results.



(a) $D = 100$ (b) $D = 500$

(a) $D = 1000$ (b) $D = 5000$

Fig. 1. Simulations of RFFKLMS (with various values of D) applied on data pairs generated by (7). The results are averaged over 100 runs. The horizontal dashed line in the figure represents the approximation of the steady-state MSE given in theorem 1.

Proposition 1. *For datasets generated by* (7) *we have:*

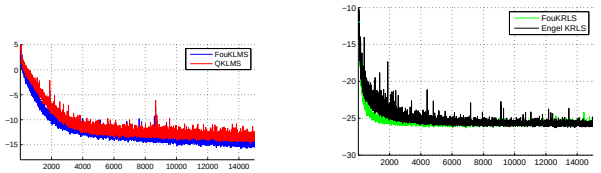
1. *If the step update parameter satisfies $0 < \mu < 2/\lambda_D$, then RFFKLSM converges in the mean, i.e., $E[\boldsymbol{\theta}_n - \boldsymbol{\theta}_{opt}] \to \mathbf{0}$.*
2. *The optimal MSE (which it is achieved when one replaces $\boldsymbol{\theta}_n$ with $\boldsymbol{\theta}_{opt}$) is given by*

   $$J_n^{opt} = \sigma_\eta^2 + E[\eta_n'] - E[\eta_n' \boldsymbol{z}_\Omega(\boldsymbol{x}_n)] R_{zz}^{-1} E[\eta_n' \boldsymbol{z}_\Omega(\boldsymbol{x}_n)^T].$$

   For large enough D, we have $J_n^{opt} \approx \sigma_\eta^2$.
3. *The excess MSE is given by $J_n^{ex} = J_n - J_n^{opt} = \text{tr}\,(R_{zz} A_n)$, where $A_n = E[(\boldsymbol{\theta}_n - \boldsymbol{\theta}_{opt})(\boldsymbol{\theta}_n - \boldsymbol{\theta}_{opt})^T]$.*
4. *If the step update parameter satisfies $0 < \mu < 1/\lambda_D$, 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 approximate the steady-state MSE ($\approx \text{tr}\,(R_{zz} A_n) + \sigma_\eta^2$).*

5. SIMULATIONS

In this Section, we present examples to illustrate the performance of the proposed algorithm and compare its behavior to the QKLMS. In all experiments, 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 ϵ 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 ϵ for which the best possible MSE (almost the same as the unsparsified version) is attained at steady state, while any smaller quantization sizes provide negligible improvements (albeit at significantly increased complexity). In all experimental setups, we tuned ϵ (using multiple trials) so that it takes a value close to this "optimal", so that to take the best possible MSE at the smallest time. 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. Table 1 gives the mean training



(a) (b)

Fig. 2. Monte Carlo simulations on data pairs generated as described in section 5.2 for (a) the RFFKLMS and QKLMS, (b) the RFFKRLS and Engel's KRLS. The results are averaged over 1000 runs.

times for QKLMS and RFFKLMS on a typical core i5 machine running Matlab (both algorithms were optimized for speed). We note that the complexity of the RFFKLMS is $\mathcal{O}(Dd)$, while the complexity of QKLMS is $\mathcal{O}(Md)$. Our experiments have shown that in order to obtain similar error floors, the required complexity of RFFKLMS is lower than that of QKLMS.

5.1. Example 1. A Linear Kernel Expansion

In this set-up we generate 5000 data pairs using (7). The input vectors $\boldsymbol{x}_n$ are drawn from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and the noise are i.i.d. Gaussian samples with $\sigma_\eta = 0.1$. The parameters of the expansion (i.e., $a_1, \ldots, a_M$) are drawn from $\mathcal{N}(0, 25)$, the kernel parameter σ is set to 5 and the step update to $\mu = 1$ (this value satisfies the requirements for convergence of Theorem 1). Figure 1 shows the evolution of the MSE for 100 realizations of the experiment. The algorithm reaches steady-state around $n = 2000$. The attained MSE is close to the approximation given in Theorem 1 (dashed line in the figure).

5.2. Example 2.

In this example, we adopt the following simple non-linear model:

$$y_n = \boldsymbol{w}_0^T \boldsymbol{x}_n + 0.1 \cdot (\boldsymbol{w}_1^T \boldsymbol{x}_n)^2 + \eta_n, \tag{9}$$

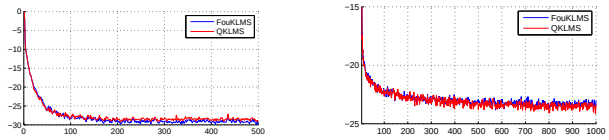
where η_n represent zero-mean i.i.d. Gaussian noise with $\sigma_\eta = 0.05$ and the coefficients of the vectors $\boldsymbol{w}_0, \boldsymbol{w}_1 \in \mathbb{R}^5$ are i.i.d. samples drawn from $\mathcal{N}(0, 1)$. Similarly to Example 1, the kernel parameter σ is set to 5 and the step update to $\mu = 1$. The quantization parameter of the QKLMS was set to $\epsilon = 5$ (leading to an average dictionary size $M = 100$) and the number of random Fourier coefficients for RFFKLMS was set to $D = 300$. Figure 2a shows the evolution of the MSE for both QKLMS and RFFKLMS running 1000 realizations of the experiment over 15000 samples.

5.3. Example 3.

Here we adopt the following chaotic series model [20]:

$$d_n = \frac{d_{n-1}}{1 + d_{n-1}^2} + u_{n-1}^3, \quad y_n = d_n + \eta_n,$$

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 and the step update to $\mu = 1$. We have also initialized d_1 to 1. Figure 3a shows the evolution of the MSE for both QKLMS and RFFKLMS running 1000 realizations of the experiment over 500 samples. The quantization parameter ϵ was set to $\epsilon = 0.01$ (leading to an average dictionary size $M = 7$), while $D = 100$.



(a) (b)

Fig. 3. Monte Carlo simulation of RFFKLMS and QKLMS applied on data pairs generated as described (a) in section 5.3 and (b) in section 5.4. The results are averaged over 1000 runs.

Experiment	QKLMS time	RFFKLMS time	QKLMS dictionary size
Example 2	0.891 sec	0.226 sec	$M = 100$
Example 3	0.036 sec	0.006 sec	$M = 7$
Example 4	0.057 sec	0.021 sec	$M = 32$

Table 1. Mean training times for QKLMS and RFFKLMS.

5.4. Example 4.

The final example adopts another chaotic series model [20]:

$$d_n = u_n + 0.5v_n - 0.2d_{n-1} + 0.35d_{n-2},$$

$$\phi(d_n) = \begin{cases} \frac{d_n}{3(0.1+0.9d_n^2)^{1/2}} & d_n \geq 0 \\ \frac{-d_n^2(1-\exp(0.7d_n))}{3} & d_n < 0 \end{cases}, \quad y_n = \phi(d_n) + \eta_n,$$

where η_n is zero-mean i.i.d. Gaussian noise with $\sigma_\eta = 0.001$, v_n is also zero-mean i.i.d. Gaussian with $\sigma_v^2 = 0.0156$ and $u_n = 0.5v_n + \hat{\eta}_n$, 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 3b shows the evolution of the MSE for both QKLMS and RFFKLMS running 1000 realizations of the experiment over 1000 samples. The parameter ϵ was set to $\epsilon = 0.01$ (leading to $M = 32$) and D was set $D = 100$.

6. THE RANDOM FOURIER FEATURES KERNEL RLS

Besides the implementation of the KLMS given in the previous sections, the rationale of the kernel approximation via random Fourier features (section 3) can also be applied to other online-algorithms such as the RLS. One only needs to choose the random samples $\boldsymbol{\omega}_i$, b_i and replace the instances of $\boldsymbol{x}_n$ in the standard RLS algorithm (see for example [4, 3]) with $\boldsymbol{z}_\Omega(\boldsymbol{x}_n)$. The resulting algorithm performs as well as the original KRLS provided by Engel [2], but it is almost twice as fast. Figure 2b compares the performances of RFFKRLS and Engle's KRLS on data samples created as in Example 5.2. The regularization parameter for the RFFKRLS was set to $\lambda = 0.0001$, the forgetting factor to $\beta = 0.9995$, while the number of random features was set to $D = 300$. The parameter for the ALD sparsification mechanism of Engel's KRLS was set to $\nu = 0.0005$.

7. CONCLUSIONS

We presented an alternative rationale for the KLMS and KRLS based on the approximation of the kernel function with random Fourier Features. The proposed algorithms exhibit similar convergence performance to the standard KLMS/KRLS algorithms, albeit they require significantly lower implementation time (due to their simplicity). Furthermore, their "linear" characteristics pave the way for generalization to other settings (e.g., the distributed KLMS [21]).

8. REFERENCES

[1] 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.

[2] 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.

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

[4] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Online learning in reproducing kernel Hilbert spaces," in *Signal Processing Theory and Machine Learning*, Rama Chellappa and Sergios Theodoridis, Eds., Academic Press Library in Signal Processing, pp. 883–987. Academic Press, 2014.

[5] 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.

[6] K. Slavakis, S. Theodoridis, and I. Yamada, "Adaptive constrained Learning in Reproducing Kernel Hilbert spaces: the robust beamforming case," *IEEE Transactions on Signal Processing*, vol. 57, no. 12, pp. 4744–4764, Dec. 2009.

[7] 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.

[8] Vaerenbergh S. V., Lázaro-Gredilla M., and Ignacio Santamaría, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Transanctions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1313–1326, Aug. 2012.

[9] 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.

[10] 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, March 2011.

[11] Badong Chen, Songlin Zhao, Pingping Zhu, and J.C. Principe, "Quantized kernel least mean square algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 22 –32, jan. 2012.

[12] C. Richard, J.C.M. 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.

[13] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering*, Hoboken, NJ: Wiley, 2010.

[14] R. Mitra and V. Bhatia, "The diffusion-klms algorithm," in *Information Technology (ICIT), 2014 International Conference on*, Dec 2014, pp. 256–259.

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

[16] Chouvardas Symeon and Draief Moez, "A diffusion kernel LMS algorithm for nonlinear adaptive networks," in *ICASSP*, 2016.

[17] Rahimi. A. and Recht B., "Random features for large scale kernel machines," in *Adv. Neural Inf. Process. Syst.* 2007, vol. 20, pp. 1177 – 1184, Vancouver, BX, Canada.

[18] Zhen Hu, Ming Lin, and Changshui Zhang, "Dependent online kernel learning with constant number of random fourier features," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2464 – 2476, October 2015.

[19] Lázaro-Gredila M., Quinonero-Candela J., Rasmussen E. C., and Figueiras-Vidal R. A., "Sparse spectrum gaussian process regression," *Journal of Machine Learning Research*, vol. 11, pp. 1865–1881, 2010.

[20] W.D. Parreira, J.C.M. 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.

[21] Pantelis Bouboulis, Simos Chouvardas, and Sergios Theodoridis, "Efficient distributed online algorithms in RKHS: A random fourier feature perspective," *submitted*.