OPERATOR-VALUED KERNEL RECURSIVE LEAST SQUARES ALGORITHM

P. O. Amblard

GIPSAlab/CNRS UMR 5283
Université de Grenoble
Grenoble, France

H. Kadri

LIF/CNRS UMR 7279
Aix-Marseille Université
Marseille, France

ABSTRACT

The paper develops recursive least square algorithms for nonlinear filtering of multivariate or functional data streams. The framework relies on kernel Hilbert spaces of operators. The results generalize to this framework the kernel recursive least squares developed in the scalar case. We particularly propose two possible extensions of the notion of approximate linear dependence of the regressors, which in the context of the paper, are operators. The development of the algorithms are done in infinite-dimensional spaces using matrices of operators. The algorithms are easily written in finite-dimensional settings using block matrices, and are illustrated in this context for the prediction of a bivariate time series.

***Index Terms*—** kernel RLS, operator-valued kernels, vector-valued RKHS, multitask learning, functional data analysis

1. INTRODUCTION

In this paper we consider the problem of online nonlinear filtering for functional data. Functional data analysis deals with applications for which data are naturally modeled as functions [1, 2, 3]. For example, this occurs in sensor networks where each sensor measures, say, the temperature over time, and the goal is to predict the whole field of temperature over time. Another relevant application could be hyperspectral images analysis, for which each pixel can be described as a whole function of the wavelength. In some application, for example ocean monitoring, functional data are available in stream. For example, a buoy in the pacific ocean regularly sends the measures it has recorded during some period. Assimilation of this functional data needs to be done online.

In this work we consider filtering using a reproducing kernel Hilbert space (RKHS) approach. This framework has been proved extremely powerful for scalar problems [4, 5] since it allows nonlinear processing using linear techniques. However, for multivariate or functional data, the usual theory of RKHS is not appropriate since it is dedicated to one dimensional output data. It would be possible to deal with product of different spaces, but this would not take correctly into account interactions between variables. This has led to the development of reproducing kernel Hilbert spaces of operators [6]. Many works have been done in this context and some applications has been developed, especially in the machine learning and statistics communities ([7, 8, 9, 10] to cite some but a few). Some work concerning the online processing of functional data using RKHS of operators are emerging, such as [11, 12].

The contributions of this paper are two operator-valued kernel recursive least square algorithms, ovkRLS, especially for multivariate and functional data filtering applications. These are developed for infinite-dimensional data, and obviously particularized for multivariate data. The algorithms rely on two different definitions of approximate linear dependence for operators.

Earlier works On-line kernel based learning has attracted a lot of attention in the the last decade, in both the machine learning and the signal processing communities. The recent review [13] gives an nice view of the field, especially for applications in filtering. In parallel and especially in the machine learning community, many works have developed the use of operator-valued kernel Hilbert spaces and their application in multitask learning and functional data analysis. In this paper, we focus on online learning with operator-valued kernels. In particular, we extend the seminal work in [14], since our results recover the usual kRLS when the outputs are scalars.

In the following section, we make precise the context of the paper and recall some necessary materials on operator-valued reproducing kernel Hilbert spaces. Section 3 is the core of the paper. We present a sparsification procedure which generalizes the approximate linear dependence procedure of the scalar case. Two possible extensions are proposed which lead to two versions of the algorithm. A simple illustration is provided in the last section. Finally, a discussion concludes the paper by highlighting remaining problems and some future developments.

2. NOTATION AND BACKGROUND

We motivate the work by a regression problem in high-dimensional spaces. Given a data set $\{x_i, y_i\}_{i=1}^N$, we look for an f so that a good model for the data is $y = f(x) + \varepsilon$. Here, we suppose that x and y belongs to some linear spaces $\mathcal{X}$ and $\mathcal{Y}$ which are not necessarily finite-dimensional (the fact that $\mathcal{X}$ is a linear space could even be relaxed.) For example, the spaces can be some functional spaces such as $L^2(I)$, I being some interval of the real line. In this case, the problem is a regression problem for functional data, and f is then an operator from $\mathcal{X}$ to $\mathcal{Y}$ which has to be inferred from the learning set $\{x_i, y_i\}_{i=1}^N$. Another important example in signal processing is the problem of prediction of a multivariate time series $\boldsymbol{z}_t = f(\boldsymbol{z}_{t-1}, \ldots, \boldsymbol{z}_{t-M}) + \varepsilon_t$. Here, if $\boldsymbol{z}_t \in \mathbb{R}^{n_z}$, the application f is an operator from $\mathbb{R}^{Mn_z}$ to $\mathbb{R}^{n_z}$ and has to be chosen so as to minimize the prediction errors.

When $\mathcal{Y}$ is $\mathbb{R}$ or $\mathbb{C}$, a powerful approach to solve these kind of problems is to look for f into a RKHS. This allows to model f as a highly nonlinear function while at the same time authorizing its estimation using very efficient linear algorithms. This has led to a lot of works in the last two decades [4]. Extending the framework when the outptut space $\mathcal{Y}$ is multidimensional or infinite-dimensional has generated some work in the last decade [7, 8, 9]. The extension considers a space of operators instead of a functional space, and this requires to switch from scalar-valued kernels to operator-valued kernels. We now formally recall the different notions we need.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

If $\mathcal{K}$ and $\mathcal{K}'$ are two Hilbert spaces, $\mathcal{L}(\mathcal{K},\mathcal{K}')$ is the Banach space of bounded linear operators from $\mathcal{K}$ to $\mathcal{K}'$. In the case of $\mathcal{K}=\mathcal{K}'$, we set $\mathcal{L}(\mathcal{K})=\mathcal{L}(\mathcal{K},\mathcal{K})$.

Hilbert spaces with reproducing operator-valued kernels [6]. Consider now a Hilbert space $\mathcal{H}$ of bounded (and hence continuous) operators from $\mathcal{X}$ to $\mathcal{Y}$, equipped with an inner product $\langle|\rangle_{\mathcal{H}}$. $\mathcal{H}$ is called a RKHS if

$$\begin{aligned}\delta_{fg}:\mathcal{H} &\longrightarrow \mathbb{R}\\ \Sigma &\longmapsto \delta_{fg}(\Sigma):=\langle g|\Sigma(f)\rangle_{\mathcal{Y}}\end{aligned}$$

is continuous $\forall f,g\in\mathcal{X}\times\mathcal{Y}$. This a generalization to operators of the continuity of the evaluation functional for functions, a defining property of usual RKHS [5]. Application of Riesz representation theorem then allows to show that there exists an operator-valued kernel $K(\cdot,\cdot):\mathcal{X}\times\mathcal{X}\to\mathcal{L}(\mathcal{Y})$ such that

$$\langle g|\Sigma(f)\rangle_{\mathcal{Y}}=\langle K(f,\cdot)g|\Sigma(\cdot)\rangle_{\mathcal{H}},\forall f,g\in\mathcal{X}\times\mathcal{Y}.$$

Note that $K(f,.)g$ is an operator from $\mathcal{X}\to\mathcal{Y}$ and belongs to $\mathcal{H}$. The preceding defining property can be seen as a reproducing property since

$$\langle g|K(f,h)k\rangle_{\mathcal{Y}}=\langle K(h,\cdot)g|K(f,\cdot)k\rangle_{\mathcal{H}}.$$

K has the following symmetry property

$$\langle K(f,h)g|k\rangle_{\mathcal{Y}}=\langle K(h,\cdot)k|K(f,\cdot)g\rangle_{\mathcal{H}}=\langle K(h,f)k|g\rangle_{\mathcal{Y}}.$$

Finally, $K(f,h),\forall f,h\in\mathcal{X}$, has a positiveness property: $\forall n\in\mathbb{N}$, $\forall\{f_i,g_i\}_{i=1}^n$

$$\begin{aligned}\sum_{i,j}\langle K(f_i,f_j)g_i|g_j\rangle_{\mathcal{Y}} &= \sum_{i,j}\langle K(f_j,\cdot)g_j|K(f_i,\cdot)g_i\rangle_{\mathcal{H}}\\ &= \|\sum_j K(f_j,\cdot)g_j\|_{\mathcal{H}}^2\geq 0.\end{aligned}$$

The properties of symmetry and positiveness are the defining properties of a kernel. We call K an operator-valued kernel on $\mathcal{X}$. It can be shown [7, 9] that a kernel determines a unique RKHS. Then most of the constructions done in the function case can be transposed in this context. In particular, the representer theorem can be proved as in [4] to show that for the problem

$$f^*=\arg\min_{f\in\mathcal{H}}\sum_i c(x_i,y_i,f(x_i))+\lambda\Omega(\|f\|_{\mathcal{H}}),$$

where c is a cost function and Ω a strictly increasing positive penalty, any minimizer can be written as $f(\cdot)=\sum_i K(x_i,\cdot)z_i, z_i\in\mathcal{Y},\ \forall i$.

3. OPERATOR-VALUED KRLS

The aim here is to learn an operator that links $x\in\mathcal{X}$ to $y\in\mathcal{Y}$ based on a data stream $\{x_i,y_i\}, i\geq 1$. We consider an operator-valued kernel K and its associated RKHS $\mathcal{H}\subset\mathcal{L}(\mathcal{X},\mathcal{Y})$. After t observations, we look for the member f_t of $\mathcal{H}$ which solves

$$f_t=\arg\min_{\Sigma\in H}\sum_{i=1}^t\|y_i-\Sigma(x_i)\|_{\mathcal{Y}}^2.$$

Thanks to the representer theorem, the minimizer has the form $\sum_i K(x_i,\cdot)z_i$, and the program becomes

$$\boldsymbol{z}_t=\arg\min_{\boldsymbol{z}\in\mathcal{Y}^t}\sum_{i=1}^t\left\|y_i-\sum_{j=1}^t K(x_j,x_i)z_j\right\|_{\mathcal{Y}}^2.$$

Working in product spaces (with the usual associated inner product) with matrices of operators allows to write this as

$$\boldsymbol{z}_t=\arg\min_{\boldsymbol{z}\in\mathcal{Y}^t}\|\boldsymbol{y}_t-\boldsymbol{K}_t\boldsymbol{z}\|_{\mathcal{Y}^t}^2.$$

where $\boldsymbol{K}_t\in\mathcal{L}(\mathcal{Y}^t)$ is a matrix of operators whose (i,j)th block is $(\boldsymbol{K}_t)_{ij}=K(x_i,x_j)$. Under some assumptions, a solution to this program can be obtained (including some regularization if needed). Two problems occur here when dealing with a stream of data. The first is how to incorporate a new observation in the learning procedure, and the second is the linear increase in dimensions of the matrix of operators $\boldsymbol{K}_t$ and of the vector of operators $\boldsymbol{y}_t$. The first problem is solved by formulating a recursive solution to the optimization problem. The second can be solved by a sparsification procedure which keeps the most representative data we have seen up to time t. Equivalently this corresponds to approximating the matrix of operators $\boldsymbol{K}_t$ by a lower dimensional matrix. This problem is well know in kernel based learning approaches for large data sets [4]. Here, since we face a data stream, a nice solution to solve both problems is to recursively build a dictionary that correctly represents the data stream. This idea was proposed in [14] to solve the scalar-valued kRLS. Having built a dictionary up to time $t-1$, the new datum x_t is incorporated in the dictionary if it can not be predicted from the members of the dictionary. This is called approximate linear dependence (ALD). We develop this idea in the operator context. It turns out that this notion for operator is not as simple as in the scalar case. We provide two possible views of ALD.

3.1. Online sparsification via ALD

Assume that at time step t, after having observed $t-1$ samples $\{x_i\}_{i=1}^{t-1}$, we have collected a dictionary consisting of a subset of the training samples $\mathcal{D}_{t-1}=\{\tilde{x}_j\}_{j=1}^{m_{t-1}}$, where by construction $\{K(\tilde{x}_j,\cdot)\}_{j=1}^{m_{t-1}}$ are linear independent feature operators. This obviously requires to define the notion of "linear independence" for feature operators. We give two possible definitions.

Linearly dependent operators. Following [7], we will say that $\{K(x_i,\cdot)\}_i$ are linearly independent, if any $y_j\in\mathcal{Y}$ can be uniquely written as $\sum_i K(x_j,x_i)c_i, c_i\in\mathcal{Y}$. Hence, when a new datum x_t is observed, we test if $K(x_t,\cdot)y$ can be written as $\sum_{i=1}^{m_{t-1}}K(\cdot,\tilde{x}_i)c_i$, and this for all y. Approximate Linear Dependence proceeds by performing a mean square prediction and by testing if the error is small or not. Let $\boldsymbol{a}=(a_1,\ldots,a_{m_{t-1}})$ where $a_j\in\mathcal{Y}$. The mean square error of prediction reads

$$\begin{aligned}\delta_t(y)&=\min_{\boldsymbol{a}}\left\|\sum_{j=1}^{m_{t-1}}K(\tilde{x}_j,\cdot)a_j-K(x_t,\cdot)y\right\|_{\mathcal{H}}^2 \qquad (1)\\ &=\min_{\boldsymbol{a}}\langle\widetilde{\boldsymbol{K}}_t\boldsymbol{a}|\boldsymbol{a}\rangle_{\mathcal{Y}^{m_{t-1}}}-2\langle\widetilde{\boldsymbol{K}}_{x_t}\boldsymbol{a}|y\rangle_{\mathcal{Y}}+\langle K(x_t,x_t)y|y\rangle_{\mathcal{Y}}.\end{aligned}$$

Here, $\widetilde{\boldsymbol{K}}_t\in\mathcal{L}(\mathcal{Y}^{m_{t-1}})$ is a matrix of operators whose (i,j)th block is $(\widetilde{\boldsymbol{K}}_t)_{ij}=K(\tilde{x}_i,\tilde{x}_j)$. Likewise, $\widetilde{\boldsymbol{K}}_{x_t}\in\mathcal{L}(\mathcal{Y}^{m_{t-1}},\mathcal{Y})$ is a row of operators whose ith block is $(\widetilde{\boldsymbol{K}}_{x_t})_i=K(x_t,\tilde{x}_i)$. Let $\boldsymbol{a}_t$ be the operator in $\mathcal{L}(\mathcal{Y}^{m_{t-1}},\mathcal{Y})$ defined by $\boldsymbol{a}_t=\widetilde{\boldsymbol{K}}_{x_t}\widetilde{\boldsymbol{K}}_t^{-1}$. The minimum is then reached at $\boldsymbol{a}_t^*(y)=\widetilde{\boldsymbol{K}}_t^{-1}\widetilde{\boldsymbol{K}}_{x_t}^*y$, where * stands for the adjoint. The ALD test consists in comparing $\delta_t(y)$ to zero (practically to a user defined threshold). The condition $\delta_t(y)>0,\forall y\in\mathcal{Y}$ is obviously tedious to verify. However, since the very meaning of ALD is to test whether the new datum x_t is worth considering in the

prediction of y_t, it seems natural to use the test with $y = y_t$. Thus, the new datum is declared ALD if

$$\delta_t(y_t) = \langle K(x_t,x_t)y_t \big| y_t\rangle_{\mathcal{Y}} - \langle \widetilde{\boldsymbol{K}}_{x_t}\boldsymbol{a}_t^*(y_t)\big|y_t\rangle_{\mathcal{Y}} \leq \delta_0,$$

where δ_0 is a user defined threshold. Note that from these equations, the scalar-valued ALD condition in [14] can be recovered by choosing $y = 1$, and does not depend on y_t.

Globally linearly dependent operators. A second choice considers that the data are dependent if $\exists c_i \in \mathbb{R}, i > 1$ not all zero such that $\sum_i c_i K(x_i,\cdot) = 0$. To implement this, when a new datum x_t is collected, we predict the operator $K(x_t,\cdot) \in \mathcal{L}(\mathcal{Y},\mathcal{H})$ using a linear combination of the operators $K(\tilde{x}_i,\cdot), i = 1,\ldots,m_{t-1}$ taken from the current dictionary. To test global linear dependence we compare

$$\delta_t = \min_{\boldsymbol{a}} \left\| \sum_{j=1}^{m_{t-1}} a_j K(\tilde{x}_j,\cdot) - K(x_t,\cdot) \right\|_{op}^2$$

to a user prescribed threshold δ_0. The norm used above is an operator norm. It can be the usual operator norm $\|K(x,\cdot)\| = \sup_y \|K(x,.)y\|_{\mathcal{H}}/\|y\|_{\mathcal{Y}}$, or the Hilbert-Schmidt norm if we suppose $K(x,\cdot)$ to be Hilbert-Schmidt. In this case, let e_k be any orthonormal system of $\mathcal{Y}$, then $\|K(x,\cdot)\|_{HS}^2 = \sum_k \|K(x,\cdot)e_k\|_{\mathcal{H}}^2$. Recall that the space $\mathcal{S}(\mathcal{Y},\mathcal{H})$ of Hilbert-Schmidt operators from $\mathcal{Y}$ to $\mathcal{H}$ is an Hilbert space when endowed with the inner product $\langle H_1, H_2\rangle := \sum_k \langle H_1 e_k \big| H_2 e_k\rangle_{\mathcal{Y}}$. δ_t then reads

$$\begin{aligned}\delta_t &= \min_{\boldsymbol{a}} \left\| \sum_{j=1}^{m_{t-1}} a_j K(\tilde{x}_j,\cdot) - K(x_t,\cdot) \right\|_{HS}^2 \\ &= \min_{\boldsymbol{a}} \sum_k \left\| \sum_{j=1}^{m_{t-1}} a_j K(\tilde{x}_j,\cdot)e_k - K(x_t,\cdot)e_k \right\|_{\mathcal{H}}^2.\end{aligned}$$

Let Tr $K = \sum_k \langle Ke_k\big|e_k\rangle_{\mathcal{Y}}$, $\widetilde{K}_{t-1}$ the matrix with entries $(\widetilde{K}_{t-1})_{ij} =$ Tr $K(\tilde{x}_i,\tilde{x}_j)$ and $\widetilde{k}_t$ the vector with entries $(\widetilde{k}_t)_i =$ Tr $K(x_t,\tilde{x}_i)$. Then we get $\delta_t = \min_{\mathbf{a}}$ Tr $K(x_t,x_t) - 2\boldsymbol{a}^\top\widetilde{k}_t + \boldsymbol{a}^\top\widetilde{K}_{t-1}\boldsymbol{a}$. The minimum is attained for $\mathbf{a}_t = \widetilde{K}_{t-1}^{-1}\widetilde{k}_t$ and hence

$$\delta_t = \text{Tr } K(x_t,x_t) - \widetilde{k}_t^\top \widetilde{K}_{t-1}^{-1}\widetilde{k}_t.$$

In the case of finite-dimensional spaces, the trace is obviously very easy to calculate. Note that if $\mathcal{Y} = \mathbb{R}$ this approach also reduces to the framework developed in [14].

3.2. Algorithm 1: okRLS

Given data up to time t, we look for the operator $f_t \in \mathcal{H}$ such that $f_t = \arg\min_{f\in\mathcal{H}} \sum_{i=1}^t \|f(x_i) - y_i\|_{\mathcal{Y}}^2$. The representer theorem for operator RKHS implies $f_t(\cdot) = \sum_i K(x_i,\cdot)z_{t,i}$, where $z_{t,i} \in \mathcal{Y}$. If the ALD sparsification technique is used, recall that $K(x_t,\cdot)y \approx \sum_{j=1}^{m_{t-1}} K(\tilde{x}_j,\cdot)a_{t,j}^*(y)$. We have for any $y \in \mathcal{Y}$ and any $j \leq t$, thanks to the reproducing property

$$\begin{aligned}\langle f_t(x_j)\big|y\rangle_{\mathcal{Y}} &= \sum_i \langle K(x_i,x_j)z_i\big|y\rangle_{\mathcal{Y}} \\ &\approx \sum_{\alpha,\beta=1}^{m_i,m_j} \langle a_{j,\beta}K(\tilde{x}_\beta,\tilde{x}_\alpha)\sum_i a_{i,\alpha}^* z_i\big|y\rangle_{\mathcal{Y}}.\end{aligned}$$

To keep simple notations, set $z_\alpha = \sum_i a_{i,\alpha}^* z_i$. Define $\boldsymbol{A}_t \in \mathcal{L}(\mathcal{Y}^{m_t},\mathcal{Y}^t)$ to be the matrix of operators whose (j,i) block is $(\boldsymbol{A}_t)_{ji} = a_{j,i}$. Note that the block $(\boldsymbol{A}_t)_{ji} = 0$ as soon as $i > m_j$. Then the column $(f_t(x_1),\ldots,f_t(x_t))^\top$ of $\mathcal{Y}^t$ equals $\boldsymbol{A}_t\widetilde{\boldsymbol{K}}_t$. Therefore, the program and its solution are

$$\boldsymbol{z}_t = \arg\min_{\boldsymbol{z}_t\in\mathcal{Y}^{m_t}} \|\boldsymbol{y}_t - \boldsymbol{A}_t\widetilde{\boldsymbol{K}}_t\boldsymbol{z}_t\|_{\mathcal{Y}^t}^2 = \widetilde{\boldsymbol{K}}_t^{-1}(\boldsymbol{A}_t^*\boldsymbol{A}_t)^{-1}\boldsymbol{A}_t^*\boldsymbol{y}_t.$$

This form can be turned into a recursive algorithm. The recursion however depends on the result of the ALD test. Suppose we have obtained $\boldsymbol{z}_{t-1}$, and that the new datum (x_t,y_t) is acquired. The derivation is done as in the scalar case. We just provide here the resulting recursion equations.

1. $K(x_t,\cdot)$ **is ALD** $\Longleftrightarrow \delta_t(y_t) \leq \delta_0$, the dictionary is unchanged, $\mathcal{D}_t = \mathcal{D}_{t-1}$, $m_t = m_{t-1}$, $\widetilde{\boldsymbol{K}}_t = \widetilde{\boldsymbol{K}}_{t-1}$. The matrices of operators $\boldsymbol{A}_t$ and $\boldsymbol{P}_t := (\boldsymbol{A}_t^*\boldsymbol{A}_t)^{-1}$ are updated *via*

$$\begin{aligned}\boldsymbol{A}_t &= \begin{pmatrix}\boldsymbol{A}_{t-1}\\ \boldsymbol{a}_t\end{pmatrix} \\ \boldsymbol{P}_t &= \boldsymbol{P}_{t-1} - \boldsymbol{P}_{t-1}\boldsymbol{a}_t^*(\boldsymbol{I} - \boldsymbol{a}_t\boldsymbol{P}_{t-1}\boldsymbol{a}_t^*)^{-1}\boldsymbol{a}_t\boldsymbol{P}_{t-1},\end{aligned}$$

and the solution $\boldsymbol{z}_t$ is updated by

$$\boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \widetilde{\boldsymbol{K}}_{t-1}^{-1}\boldsymbol{P}_{t-1}\boldsymbol{a}_t^*(\boldsymbol{I} - \boldsymbol{a}_t\boldsymbol{P}_{t-1}\boldsymbol{a}_t^*)^{-1}(y_t - \widetilde{\boldsymbol{K}}_{x_t}\boldsymbol{z}_{t-1}).$$

2. $K(x_t,\cdot)$ **is not ALD** $\Longleftrightarrow \delta_t(y_t) > \delta_0$, the dictionary is updated as $\mathcal{D}_t = \mathcal{D}_{t-1} \cup x_t$, $m_t = m_{t-1} + 1$, the matrices of operators $\boldsymbol{A}_t$ and $\boldsymbol{P}_t$ are updated by ($\boldsymbol{I}$ is the identity operator)

$$\boldsymbol{A}_t = \begin{pmatrix}\boldsymbol{A}_{t-1} & 0\\ 0 & \boldsymbol{I}\end{pmatrix} \text{ and } \boldsymbol{P}_t = \begin{pmatrix}\boldsymbol{P}_{t-1} & 0\\ 0 & \boldsymbol{I}\end{pmatrix}.$$

The matrix of operators $\widetilde{\boldsymbol{K}}_t^{-1}$ can be updated *via*

$$\begin{aligned}\widetilde{\boldsymbol{K}}_t^{-1} &= \left(\begin{array}{c|c}\widetilde{\boldsymbol{K}}_{t-1} & \widetilde{\boldsymbol{K}}_{x_t}^*\\ \hline \widetilde{\boldsymbol{K}}_{x_t} & K(x_t,x_t)\end{array}\right)^{-1} \\ &= \left(\begin{array}{c|c}\widetilde{\boldsymbol{K}}_{t-1}^{-1} + \widetilde{\boldsymbol{K}}_{t-1}^{-1}\widetilde{\boldsymbol{K}}_{x_t}^*\boldsymbol{Z}_t\widetilde{\boldsymbol{K}}_{x_t}\widetilde{\boldsymbol{K}}_{t-1}^{-1} & -\widetilde{\boldsymbol{K}}_{t-1}^{-1}\widetilde{\boldsymbol{K}}_{x_t}^*\boldsymbol{Z}_t\\ \hline -\boldsymbol{Z}_t\widetilde{\boldsymbol{K}}_{x_t}\widetilde{\boldsymbol{K}}_{t-1}^{-1} & \boldsymbol{Z}_t\end{array}\right),\end{aligned}$$

where $\boldsymbol{Z}_t = (K(x_t,x_t) - \widetilde{\boldsymbol{K}}_{x_t}\widetilde{\boldsymbol{K}}_{t-1}^{-1}\widetilde{\boldsymbol{K}}_{x_t}^*)^{-1}$. The vector $\boldsymbol{z}_t$ is updated by the recursion

$$\boldsymbol{z}_t = \begin{pmatrix}\boldsymbol{z}_{t-1} - \widetilde{\boldsymbol{K}}_{t-1}^{-1}\widetilde{\boldsymbol{K}}_{x_t}^*\boldsymbol{Z}_t\left(y_t - \widetilde{\boldsymbol{K}}_{x_t}\boldsymbol{z}_{t-1}\right)\\ \boldsymbol{Z}_t\left(y_t - \widetilde{\boldsymbol{K}}_{x_t}\boldsymbol{z}_{t-1}\right)\end{pmatrix}.$$

To sum up the algorithm, for each $t \geq 1$, first evaluate $\boldsymbol{a}_t$ and δ_t, and apply item 1 above if ALD is true ($\delta_t \leq \delta_0$) or item 2 if not.

3.3. Algorithm 2: Global okRLS

When the global ALD condition is used, the structure of the algorithm is a little bit altered. For the sake of illustration, we present it in the finite-dimensional setting for which $\mathcal{Y} = \mathbb{R}^{n_y}$. In this situation, the matrix $\boldsymbol{K}_t$ is approximated by $\boldsymbol{K}_t \approx (\boldsymbol{A}_t \otimes \boldsymbol{I}_{n_y})\widetilde{\boldsymbol{K}}_t(\boldsymbol{A}_t \otimes \boldsymbol{I}_{n_y})^\top$, where $\boldsymbol{K}_t$ is the $n_y t \times n_y t$ block matrix whose (s,t)th block is the $n_y \times n_y$ matrix $K(x_s,x_t)$, where $\boldsymbol{I}_{n_y}$ is the n_y dimensional identity matrix, and $\otimes$ stands for the Kronecker product. Likewise, $\widetilde{\boldsymbol{K}}_t$ is the $m_t n_y \times m_t n_y$ block matrix with defining blocks $K(\tilde{x}_i,\tilde{x}_j)$. $\boldsymbol{A}_t$ is the $t \times m_t$ $(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_t)^\top$ where $\boldsymbol{a}_i$ is obtained by

the global ALD criterion. When sparsification based on the global ALD is used, we have

$$\begin{aligned} \boldsymbol{z}_t &= \arg\min_{\boldsymbol{z}} \left\| \boldsymbol{y}_t - (\boldsymbol{A}_t \otimes \boldsymbol{I}_{n_y}) \widetilde{\boldsymbol{K}}_t \boldsymbol{z} \right\|^2 \\ &= \widetilde{\boldsymbol{K}}_t^{-1} \left[(\boldsymbol{A}_t^\top \boldsymbol{A}_t)^{-1} \boldsymbol{A}_t^\top \right] \otimes \boldsymbol{I}_{n_y} \boldsymbol{y}_t. \end{aligned}$$

This form is easily turned into a recursive formula, along the lines developed previously. When the new datum is not ALD, the structure of the algorithm is the same as in algorithm 1. However, the structure is a little bit altered when the new datum is ALD. In that case, $\boldsymbol{P}_t = (\boldsymbol{A}_t^\top \boldsymbol{A}_t)^{-1}$ and $\boldsymbol{z}_t$ are updated according to

$$\boldsymbol{q}_t = \frac{\boldsymbol{P}_{t-1}\boldsymbol{a}_t}{1 + \boldsymbol{a}_t^\top \boldsymbol{P}_{t-1} \boldsymbol{a}_t}, \qquad \boldsymbol{P}_t = \boldsymbol{P}_{t-1} - \boldsymbol{q}_t \boldsymbol{a}_t^\top \boldsymbol{P}_{t-1},$$
$$\boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \widetilde{\boldsymbol{K}}_{t-1}^{-1} \boldsymbol{q}_t \left(y_t - (\boldsymbol{a}_t^\top \otimes \boldsymbol{I}_{n_y}) \widetilde{\boldsymbol{K}}_{t-1} \boldsymbol{z}_{t-1} \right).$$

If $n_y = 1$ it is easy to verify that algorithm 2 reduces also to the scalar-valued kRLS [14].

Complexity Analysis. A standard implementation of okRLS and Global okRLS requires $O(n_y^3 m_t^2)$ operations when $\dim \mathcal{Y} = n_y < \infty$. Thus, okRLS algorithms significantly alleviate the computational bottleneck of batch operator-valued least squares, which takes $O(n_y^3 t^3)$ time. The computational cost of both okRLS algorithms is dominated by the cost of finding the matrix $\boldsymbol{Z}_t$ used to compute the update of $\widetilde{\boldsymbol{K}}_t^{-1}$ in the non-ALD case. However, it is important to note that the cost of evaluating the ALD condition in the Global okRLS algorithm is much less than that of okRLS. For the former the total number of iterations is of $O(n_y m_t^2)$, whereas for the latter it is of $O(n_y^3 m_t^2)$. While ALD involves the multiplication of block matrices of size $n_y m_t \times n_y m_t$ and $n_y m_t \times n_y$, global ALD only involves computing the trace of the operator outputted by the kernel and the multiplication of simple matrices of size $m_t \times m_t$ and $m_t \times 1$.

4. AN ILLUSTRATION

We perform a simple experiment to illustrate the okRLS algorithms. By no means this small experiment is intended to be a complete validation of the algorithms. The numerical stability of the algorithms has to be established, especially because the okRLS are intended to be applied to functional data for which the dimensions will be high. But this is beyond the scope of the paper.

We apply the algorithms to the nonlinear prediction of a multivariate time series. We consider the coupling of Glass-Mackey like models. The bivariate time series is defined by

$$x_t = x_{t-1} - 0.4\left(x_{t-1} - \frac{2x_{t-4}}{1 + x_{t-4}^{10}}\right) y_{t-5} + 0.3 y_{t-3} + \varepsilon_{x,t}$$
$$y_t = 0.6 y_{t-1} + \frac{0.8 y_{t-2}}{1 + y_{t-2}^{10}} + 0.4 x_{t-2} + \varepsilon_{y,t}$$

where $\varepsilon_{x,t}$ and $\varepsilon_{y,t}$ are i.i.d. zero-mean Gaussian noises of variance 10^{-2}. Based on the observation of the time series up to time t, the aim is to estimate (x_{t+1}, y_{t+1}) online using okRLS. The kernel used here is the simple separable kernel operator $K(x, y) = k(x, y)T$ where k is a scalar kernel on $\mathcal{X}$ (we chose the Gaussian kernel in our simulations) and T is a constant operator. A simple choice for T is the identity operator. In this case however, the okRLS corresponds to the evaluation of scalar-valued kRLS in parallel. Furthermore, this choice does not allow to use the interactions that may exit between the components of the output (here, correlations between x_{t+1} and y_{t+1}.) A interesting choice to take these interactions into account is to make T dependent on the output. A simple way of proceeding is to choose T as the covariance of the output. This is easy to implement since the covariance can be learnt online.

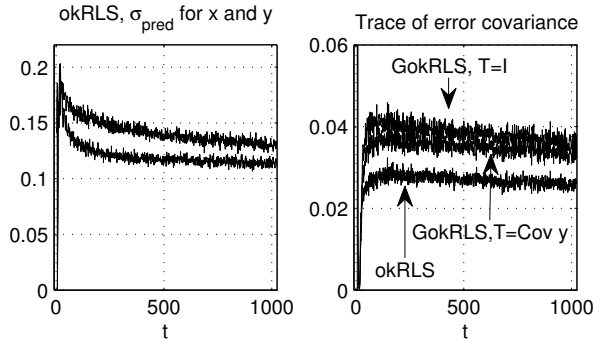

Fig. 1. Left: Standard deviation curves for the okRLS for each components of the bivariate process. Right: Trace of the covariance of the error of prediction of the bivariate process for the okRLS and GokRLS with $T = \text{cov output}$, and GokRLS with $T = I$.

The data used are $\{\boldsymbol{x}_i = (x_t, y_t)_{t=i-4}^{i}; \boldsymbol{y}_i = (x_{i+1}, y_{i+1})\}_{i \geq 5}$. We plot in figure 1, left, the standard deviation of the error of prediction $z_{t+1} - \hat{z}_{t+1}$, where $\hat{z}_{t+1}$ is the estimation by the online algorithm for each component $z = x$ and $z = y$. The algorithm is the okRLS with T being the covariance of the output as described above. In the right plot, we depict the trace of the covariance of the error of prediction for the okRLS and GokRLS with T as above, and for the GokRLS with $T = I$. This last case corresponds to scalar-valued kRLS in parallel. For both plot we only depict the 1024 first iterations. The parameters (variance of the Gaussian kernel, threshold of the ALD test) of each algorithm were chosen so that they use on average $m_t = 60$ regressors in the dictionary after 1024 iterations. The curves are obtained by averaging over 1000 independent snapshots.

As can be seen, convergence has not completly settled but is almost reached after 1024 iterations. Note that in the left plot, the lower bound for the curves are 0.1 which is the standard deviations of the dynamical noise chosen in the model. The convergence of the okRLS with the chosen kernel is thus satisfactory. The right plot allows to quantify the difference between okRLS and GokRLS. The gain is significant for the okRLS at the expense of a higher computational complexity. The plot allows also to show the improvement when considering the covariance operator of the output as T instead of I, as illustrated with the GokRLS for the two cases.

Discussion. The simple example developed previously shows the importance of coupling the outputs *via* the choice of the kernel. This coupling clearly improves prediction over the application of kRLS in parallel ($T = I$). However, this example is very simple and we are currently developping examples in much higher dimensions for functional valued signals.

Finally, we have not elucidated relashionships between the two forms of ALD we have studied, a question of great theoretical and practical interest. Concerning future works, we intend to develop a sliding ovkRLS and an ovkLMS, study the convergence of the algorithms, and apply all these algorithms to Granger causality in high dimensional spaces.

5. REFERENCES

[1] J. O. Ramsay and B. W. Silverman, *Applied functional data analysis: Methods and case studies*, Springer, 2002.

[2] F. Ferraty and P. Vieu, *Nonparametric Functional Data Analysis*, Springer Verlag, 2006.

[3] D. Bosq and D. Blanke, *Inference and prediction in large dimensions*, John Wiley & Sons: Chichester, UK, 2007.

[4] B. Schölkopf and A. J. Smola, *Learning with kernels*, MIT Press, Cambridge, Ma, USA, 2002.

[5] I. Steinwart and A. Christmann, *Support vector machines*, Springer, 2008.

[6] E. Senkene and A. Tempel'man, "Hilbert spaces of operator-valued functions," *Lithuanian Mathematical Journal*, , no. 4, pp. 665–670, 1973.

[7] C. A. Micchelli and M. Pontil, "On learning vector-valued functions," *Neural Computation*, vol. 17, pp. 177–204, 2005.

[8] H. Lian, "Nonlinear functional models for functional responses in reproducing kernel Hilbert spaces," *The Canadian Journal of Statistics*, vol. 35, pp. 597–606, 2007.

[9] H. Kadri, E. Duflos, P. Preux, S. Canu, and M. Davy, "Nonlinear functional regression: a functional RKHS approach," in *Y.W. Teh and M. Titterington (Eds.), Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR: W&CP 9*, Chia Laguna, Sardinia, Italy, 2010, pp. 111–125.

[10] N. Lim, F. d'Alché Buc, F. Auliac, and G. Michailidis, "Operator-valued kernel-based vector autoregressive models for network inference," *Machine learning*, vol. 99, no. 3, pp. 489–513, 2014.

[11] G. Pillonetto, F. Dinuzzo, and G. De Nicolao, "Bayesian online multitask learning of gaussian processes," *IEEE Trans. on PAMI*, vol. 32, no. 2, pp. 193—205, 2010.

[12] J. Audiffren and H. Kadri, "Online learning with operator-valued kernels," in *Proceedings of the 23th Symposium on Artificial Neural Networks (ESANN)*, 2015.

[13] K. Slavakis, P. Bouboulis, and S. Theodoridis, *Academic Press Library in Signal Processing: Volume 1, Signal Processing Theory and Machine Learning*, chapter ch. 17, Online learning in reproducing kernel Hilbert spaces, pp. 883–987, Elsevier, 2014.

[14] Yaakov Engel, Shie Mannor, and Ron Meir, "The kernel recursive least squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2284, 2004.