Online kernel principal component analysis: a reduced-order model

Paul Honeine

Online kernel principal component analysis: a reduced-order model

Paul Honeine, *Member, IEEE*

Abstract—Kernel principal component analysis (kernel-PCA) is an elegant nonlinear extension of one of the mostly used data analysis and dimensionality reduction techniques, the principal component analysis. In this paper, we propose an online algorithm for kernel-PCA. To this end, we examine a kernel-based version of Oja's rule, initially put forward to extract a linear principal axe. As with most kernel-based machines, the model order equals the number of available observations. To provide an online scheme, we propose to control the model order. We discuss theoretical results, such as an upper bound on the error of approximating the principal functions with the reduced-order model. We derive a recursive algorithm to discover the first principal axe, and extend it to multiple axes. Experimental results demonstrate the effectiveness of the proposed approach, both on synthetic dataset and on images of handwritten digits, with comparison to classical kernel-PCA and iterative kernel-PCA.

Index Terms—Principal component analysis, online algorithm, machine learning, reproducing kernel, Oja's rule, recursive algorithm.

1 Introduction

Principal component analysis (PCA) is a powerful tool for data analysis and dimensionality reduction [1]. It consists principally in determining a subspace that explains most of the variance of the data. By projecting data into this subspace, one can operate a dimensionality reduction procedure and extract the structure of the data. This is classically achieved by diagonalizing the covariance matrix of the data, as its eigenvectors associated to the largest eigenvalues correspond to the principal axes defining this subspace. With a computational complexity of order three with the size of the dataset, this may become cumbersome for largescale datasets, often required in order to retrieve principal axes of complex data distributions in high dimensional spaces. To address this drawback, different incremental techniques have been proposed, such as [2], [3], or more recently [4] (and references therein), however still with high computational cost. Adapted for online learning, artificial neural network methods gained popularity for addressing this problem, propelled by the work of Oja for the first principal axe determination [5], [6] and its extension by Sanger for multiple axes [7], [8]; the latter is often known as the generalized Hebbian learning.

Initiated by the pioneering work of Aronszajn [9], the theory behind reproducing kernel Hilbert space (RKHS) provided in the last decade new advances in machine learning. Moreover, it offers an elegant framework to derive nonlinear techniques based on conventional linear ones, as long as all operations can be expressed only in terms of inner products of the data. This is initially motivated by the fundamental work of Vapnik on support vector machines (SVM) for regression and classification [10], even though the concept of *kernel trick* was first published by Aizerman *et al.* in [11]. Developed in the last decade, an armada of techniques takes advantage of this concept, such as kernel Fisher discriminant analysis [12], kernel partial least squares regression [13], kernel basis pursuit [14], to name just a few (see for instance [15] for a survey of kernel methods for pattern recognition). Based on this concept, Schölkopf *et al.* [16] propose a nonlinear counterpart of PCA, the kernel-PCA. More recently, an iterative technique for kernel-PCA is elaborated by Kim *et al.* in [17], [18] by kernelizing Oja's and Sanger's rules. This is known as the kernel Hebbian algorithm for iterative kernel-PCA. Still, the number of available observations should be fixed in advance, and they are assumed to be known in advance.

Traditional kernel methods are essentially *batch* optimization problems, with all training data available in advance. Such techniques are unsuitable for largescale datasets, and thus unadapted for real-time applications. To address these drawbacks, online learning in kernel machines attracted a lot of interest in the last couple of years. Online learning refers to a paradigm where, at each time instant, a new observation is available, and the model needs to be updated according to it without having to re-explore all previously available data. Most of the work in that direction has been focused on classification and regression problems [19], [20], [21], [22], [23]; only a few attempts have been made for unsupervised learning. While Oja's classical rule is initially proposed for online PCA, its kernelized counterpart is not adapted such task, since it operates iteratively on batch datasets, while their number should be finite and fixed in advance [17]. Unfortunately, this drawback is expected in most kernel machines, since the order of the model for the optimal solution corresponds to the number of training data. This is a consequence of the well-known Representer Theorem [24], [25]. Its application to the kernel-

• *M. Honeine is with the Institut Charles Delaunay (UMR CNRS 6279), Laboratoire de Modélisation et Sûreté des Systèmes, Université de Technologie de Troyes, 10000 Troyes, France.*

PCA results in the fact that each principal axe can be expressed as a linear combination of the kernel functions associated to the available training data. Therefore, the number of available observations determines the model order, *i.e.*, number of elements in the linear combination.

In order to overcome this problem and derive an online kernel-PCA algorithm, we propose in this paper to control online the order of the model. Since the principal axes lie within the span by the kernel functions of the training data in the RKHS, we suggest to restrain this span to some selected kernel functions. Upon arrival of a newly available observation, a selection criterion determines if the corresponding kernel function either can be discarded from the model, or should be added to it. In the latter case, this operation will increase the model order, thus augmenting the spread in the RKHS. We shall henceforth refer to the selection criterion as the order control criterion. We examine a distance-based selection criterion often considered for sparse regression [26], [27]. Underlying this strategy, it turns out that the reduced-order model can efficiently approximate the optimal principal axes with optimality in the sense of solving kernel-PCA over the whole dataset. To this end, we study the approximation error and derive an upper bound which is inversely proportional to the corresponding eigenvalue, which means that, one can approximate with small errors principal axes associated to high eigenvalues, *i.e.*, most relevant principal axes. Experimental results demonstrate the relevance of the theoretical study.

The remainder of this paper is organized as follows. We begin in Section 2 by reviewing briefly Oja's rule for PCA analysis, and show its inadaptability for online kernel-PCA learning. Section 3 is devoted to studying the order control criterion, as we study some properties of the resulting model, including approximation errors. In Section 4, we derive the online kernel-PCA algorithm, and study in Section 5 some well-known issues such as the rate of convergence, centering/noncentering the data and the the denoising scheme with reduced-order models. We illustrate the relevance of the proposed method with experimentations on both synthetic and real datasets in Section 6. But before, we prepare the grounds by giving some connections relating PCA literature with our work.

Related (and unrelated) work

The proposed method is an online algorithm for kernel-PCA. In its traditional linear version, PCA can be solved with offline batch algorithms [1], as well as online learning algorithms with Oja's [5] and Sanger's [7] rules (also known as generalized Hebbian algorithms or GHA). Kernel-PCA [16] and the kernel Hebbian algorithms [18] are the nonlinear extensions of these algorithms, by embedding data into a high dimensional feature space. As with kernel machines, they are build on models, linear in the feature space, with an order equal to the number of available data. Therefore, these kernel machines are in essence offline algorithms, in batch setting for kernel-PCA and iterative mode for kernel Hebbian algorithm (kHA) for iterative kernel-PCA. The number of available data should be fixed in advance, and they are assumed to be known in advance. Moreover, to converge to the PCA solution, the iterative kernel-PCA requires many passes through the entire dataset.

TABLE 1
Comparison between several PCA algorithms

	Model	Setting
Principal component analysis [1]	linear	batch
Kernel-PCA (kernel machines) [16]	nonlinear	batch
Oja's [5] and Sanger's [7] rules (GHA)	linear	online
Iterative kernel-PCA (KHA) [18]	nonlinear	iterative
Online kernel-PCA [this paper]	**nonlinear**	**online**

In this paper, we focus on the online learning scenario, which consists of a potentially infinite stream of observations, presented one at a time (a single pass is available). Such online learning algorithm has space requirements that are independent on the number of data, and we do not need to store the entire training dataset in memory. Table 1 gives connections and differences with other PCA techniques.

The proposed approach can be regarded as jointly a sparsification technique followed by a recursive updating scheme, both adapted for online processing. Thus it is related to sparse techniques for kernel-PCA (beyond naive random selection), but often inconvenient for online learning. For instance, sparsification technique based on [26] are studied in [28] to derive a greedy spectral embedding algorithm. However, it requires a matrix inversion and an eigen-decomposition at each iteration, rendering it computational expensive and thus untractable for online learning. The sparse kernel-PCA algorithm proposed in [29] approximates the covariance matrix from a (weighted) subset of available data. The subset is determined by a maximum likelihood criterion, based on a probabilistic formulation of PCA. Such approach suffers from many drawbacks, mainly for relying on a probabilistic model for the data. Moreover, an online version of this technique seems to be impractical. This is also the case for ℓ_1-norm penalization with high computational cost. To circumvent this difficulty, the authors of [30], [31] derive a projection pursuit scheme to maximize a contrast function, rather than the variance and thereby dropping the concept of principal components.

Before proceeding, an important issue needs to be clarified. In classical PCA, one often discards some dimensions of the data, in order to improve the interpretability of the results. This is the essence of principal variables selection [32]. In this spirit, a sparse PCA is derived either by incorporating an ℓ_1-norm penalization in the classical formulation [33], or by solving a semidefinite programming problem [34]. This is fundamentally differ-

ent from our approach (and the ones described above), as we seek representatives that describe well the data (samples), and not determining the redundant features or proceeding in a feature selection purpose.

2 PCA AND OJA'S RULE

Let $\mathcal{X} \subset \mathbb{R}^p$ be a vector space, with the conventional inner product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \boldsymbol{x}_i^\top \boldsymbol{x}_j$ for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$. The (orthogonal) projection of any $\boldsymbol{x} \in \mathcal{X}$ onto some vector $\boldsymbol{w} \in \mathcal{X}$ is given by the real-value $y = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \boldsymbol{w}^\top \boldsymbol{x}$ and the direction (or axe) defined by $\boldsymbol{w}$. Conventional PCA seeks the axe that captures most of the variance of the data. This is obtained by solving the eigen-decomposition problem $C\boldsymbol{w} = \lambda \boldsymbol{w}$, where $C = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i^\top$ is the covariance of the data (assumed centered). The optimal principal axe, denoted $\boldsymbol{w}^*$, is given by the eigenvector associated to the largest eigenvalue. Since C is a p-by-p matrix, the computational complexity of such operation is $\mathcal{O}(p^3)$.

Consider a set of observations $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \ldots\}$, with $\boldsymbol{x}_t$ available at time instant t. Oja proposes in [5] to learn iteratively the first principal axe with the updating rule

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t(\boldsymbol{x}_t y_t - y_t^2 \boldsymbol{w}_t), \tag{1}$$

where η_t is the stepsize parameter and $y_t = \boldsymbol{w}_t^\top \boldsymbol{x}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_t$. By examining the incremental change in this expression, the term $\boldsymbol{x}_t y_t$ leads to the vector that maximizes the projection, while the second term constrains its norm. This learning rule converges to the first principal axe $\boldsymbol{w}^*$. To prove this, we observe that when $\boldsymbol{w}_t$ converges to some state $\boldsymbol{w}$, we have $\boldsymbol{x}_t y_t = y_t^2 \boldsymbol{w}$, or equivalently $\boldsymbol{x}_t \boldsymbol{x}_t^\top \boldsymbol{w} = \boldsymbol{w}^\top \boldsymbol{x}_t \boldsymbol{x}_t^\top \boldsymbol{w}\boldsymbol{w}$. Averaging over the whole data, we get the expression $C\boldsymbol{w} = \boldsymbol{w}^\top C\boldsymbol{w}\, \boldsymbol{w}$. This is the well-known eigen-decomposition problem of the covariance matrix, with the eigenvalue $\boldsymbol{w}^\top C\boldsymbol{w}$ corresponding to the squared output y that one wishes to maximize. Therefore the resulting vector from (1) converges to the largest eigenvector of C, namely $\boldsymbol{w}^*$.

Now let us apply these techniques in an RKHS, leading to nonlinear PCA analysis. Let $\mathcal{H}$ be the RKHS induced by the reproducing kernel $\kappa(\cdot,\cdot)$, and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ its inner product. This means that for all functions $\psi(\cdot) \in \mathcal{H}$, we have the evaluation property $\psi(\boldsymbol{x}) = \langle \psi(\cdot), \kappa(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}}$ for any $\boldsymbol{x} \in \mathcal{X}$, leading to the reproducing property $\langle \kappa(\boldsymbol{x}_i, \cdot), \kappa(\boldsymbol{x}_j, \cdot) \rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Examples of kernel functions include

- the Gaussian kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2)$
- the exponential kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\| / \sigma)$
- and the quadratic kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = |\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1|^2$,

where σ is a positive parameter defining the kernel bandwidth.

By representing each $\boldsymbol{x}$ of $\mathcal{X}$ by a kernel function $\kappa(\boldsymbol{x}, \cdot)$ in $\mathcal{H}$, one can apply PCA techniques in the latter space, to the n kernel functions, $\kappa(\boldsymbol{x}_1, \cdot), \kappa(\boldsymbol{x}_2, \cdot), \ldots, \kappa(\boldsymbol{x}_n, \cdot)$. As proved by Schölkopf *et al.* [16], the principal axes lie into the span of of the kernel functions associated to the available data. In other words, these principal axes (or principal functions to be more precise since we are working in a functional space, RKHS) take the form

$$\psi(\cdot) = \sum_{k=1}^n \alpha_k \, \kappa(\boldsymbol{x}_k, \cdot), \tag{2}$$

for n available observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$. Thus the projection of $\kappa(\boldsymbol{x}, \cdot)$ onto $\psi(\cdot)$ is given by

$$\psi(\boldsymbol{x}) = \langle \psi(\cdot), \kappa(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}} = \sum_{k=1}^n \alpha_k \, \kappa(\boldsymbol{x}_k, \boldsymbol{x}),$$

where we use both the evaluation and the reproducing properties. Without getting into details[1], the optimal coefficients $\alpha_1, \alpha_2, \ldots \alpha_n$, for any arbitrary principal function, are obtained by diagonalizing the n-by-n so-called Gram matrix $\boldsymbol{K}$ whose (i,j)-th entry is $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. These coefficients are normalized such that

$$\sum_{k=1}^n \alpha_k^2 = \frac{1}{n\lambda_r}, \tag{3}$$

where λ_r is the corresponding eigenvalue of the matrix C. The computational complexity to solve such eigen-decomposition problem is $\mathcal{O}(n^3)$, which can be reduced to $\mathcal{O}(n^2)$ with recursive techniques, however still unadapted for online learning.

In order to kernelize Oja's rule, one wishes to apply it to the kernel functions in the RKHS. By operating the update rule (1) in $\mathcal{H}$, the principal function at time instant t is given by the expression

$$\psi_{t+1}(\cdot) = \psi_t(\cdot) + \eta_t \left(y_t \kappa(\boldsymbol{x}_t, \cdot) - y_t^2 \, \psi_t(\cdot) \right), \tag{4}$$

where $y_t = \psi_t(\boldsymbol{x}_t)$ is the value of the projection of $\kappa(\boldsymbol{x}_t, \cdot)$ onto $\psi_t(\cdot)$. This is the essence of the iterative approach proposed by Kim *et al.* in [17], [18], where the model (2) is considered, namely at instant t

$$\psi_t(\cdot) = \sum_{k=1}^n \alpha_{k,t} \, \kappa(\boldsymbol{x}_k, \cdot). \tag{5}$$

By injecting this model in (4) and from $y_t = \psi_t(\boldsymbol{x}_t)$, we get an update rule of the coefficients $\alpha_{1,t}, \alpha_{2,t}, \ldots, \alpha_{n,t}$, with

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_t \, y_t (\boldsymbol{\beta}_t - y_t \, \boldsymbol{\alpha}_t), \tag{6}$$

where $\boldsymbol{\alpha}_t = [\alpha_{1,t} \; \alpha_{2,t} \; \cdots \; \alpha_{n,t}]^\top$. In this expression, $\boldsymbol{\beta}_t = [0 \, 0 \; \cdots \; 0 \, 1 \, 0 \; \cdots \; 0 \, 0]^\top$ is the sparse n-by-1 column vector of zeros except for the $(t \bmod n)$-th entry which is set to 1. To achieve convergence, it is put forward in [17], [18] that this operation needs to be repeated, up to 800 times, over the entire available dataset, randomly permuted at each sweep.

This algorithm clearly needs not to evaluate the Gram matrix, and is therefore adapted for largescale training datasets. However, as illustrated in the updating rule (6),

1. See Section 5.2 for a discussion on centering the data in the feature space.

$\boldsymbol{\alpha}_{t+1}$ and $\boldsymbol{\beta}_t$ are vectors of n entries, n being the number of observations. Therefore, the number of observations must be known in advance, and of fixed size, leading to an iterative technique unadapted for online learning, i.e., infinite stream of data. This is due to the fact that the model order corresponds to the number of available data, namely n. To circumvent this drawback, one has to control the order of the model, as we propose in the following section.

3 Selection Criterion for Subspace Control

Reducing the model order is a very active research area within the community of researchers in kernel machines [35], as well as Gaussian processes [36] [37, Chapter 4], since the underlying models of these methods have the form (2) as proven by the well-known Representer Theorem [25]. Thus, the computational cost of evaluating this model (on a new observation) is linear with the number of available training data. The popularity of SVM is in part ascribed to cut down this burden, since the resulting model is constituted of only a small fraction of the training data, the *support vectors*. While this still remains expensive for largescale datasets compared to neural networks, bringing down additionally the model order may often be required. Many methods are elaborated for this purpose, following the work initiated by Burges [38] as well as the wide literature on sparse representations (see for instance [39], [40]). For instance, Downs *et al.* propose in [41] a pruning technique, by removing kernel functions if they can be approximated by a linear combination of the remaining ones. In [42], incremental techniques are derived for controlling the complexity in SVM for classification. For this task, a selection criterion is considered to determine at each instant if the newly available kernel function must be included to the model, and thus incrementing its order, or it is discarded. Various criteria for quantifying the relevance of a kernel function in SVM are put forward by Keerthi *et al.* in the more recent work [43]. Most selection criteria are inappropriate for the online kernel-PCA, since they are either computational expensive and thus unadapted for online learning, or they are often applied to supervised learning with training data consisting of input-output couples, by roughly solving a cost functional in order to determine the resulting improvement. In this paper, we examine a distance-based selection criterion, initially investigated for sparse regression [26], [27], and we show its appropriateness for the online kernel-PCA algorithm.

We propose a m-order model for the (first) principal function at instant t, with[2]

$$\psi_t(\cdot) = \sum_{k=1}^{m} \alpha_{k,t}\, \kappa(\boldsymbol{x}_{\omega_k}, \cdot), \tag{7}$$

2. With a slight abuse of notation, the order is denoted m, while it depends on time t, and thus one should read $m(t)$.

where the m kernel functions $\kappa(\boldsymbol{x}_{\omega_k}, \cdot)$ are selected from the t kernel functions available so far, namely $\{\omega_1, \omega_2, \ldots, \omega_m\} \subset \{1, 2, \ldots, t\}$. At each instant t, upon arrival of $\boldsymbol{x}_t$, we consider a selection criterion for adding the kernel function $\kappa(\boldsymbol{x}_t, \cdot)$ to the expansion of $\psi_t(\cdot)$, thus incrementing its order. Next, we derive the selection criterion, and study properties of the resulting model, such as an upper bound on the error of approximating the exact, full-order, principal functions.

3.1 The subspace control criterion

The reduced-order model (7) defines a subspace, spanned by the m kernel functions. The error of approximating any element $\kappa(\boldsymbol{x}_t, \cdot)$ by this model is given by the norm of the residual error function. Let $\boldsymbol{P}_m$ denotes the projection operator onto the subspace spanned by $\kappa(\boldsymbol{x}_{\omega_1}, \cdot), \kappa(\boldsymbol{x}_{\omega_2}, \cdot), \ldots, \kappa(\boldsymbol{x}_{\omega_m}, \cdot)$. Then the (squared) approximation error of $\kappa(\boldsymbol{x}_t, \cdot)$ by a linear combination of these kernel functions is given by

$$\epsilon_t = \left\| (\boldsymbol{I} - \boldsymbol{P}_m)\kappa(\boldsymbol{x}_t, \cdot) \right\|_{\mathcal{H}}^2,$$

where $\boldsymbol{I}$ is the identity operator. Let $\sum_{k=1}^{m} \beta_k\, \kappa(\boldsymbol{x}_{\omega_k}, \cdot)$ denotes the projection of $\kappa(\boldsymbol{x}_t, \cdot)$ onto this subspace, namely $\boldsymbol{P}_m \kappa(\boldsymbol{x}_t, \cdot)$, then

$$\epsilon_t = \min_{\boldsymbol{\beta}} \left\| \kappa(\boldsymbol{x}_t, \cdot) - \sum_{k=1}^{m} \beta_k\, \kappa(\boldsymbol{x}_{\omega_k}, \cdot) \right\|_{\mathcal{H}}^2.$$

By expanding this norm, we get a matrix notation in terms of $\boldsymbol{\beta} = [\beta_1\ \beta_2 \cdots \beta_m]^\top$, with

$$\begin{aligned}\epsilon_t &= \min_{\boldsymbol{\beta}} \sum_{k,l=1}^{m} \beta_k \beta_l \kappa(\boldsymbol{x}_{\omega_k}, \boldsymbol{x}_{\omega_l}) - 2\sum_{k=1}^{m} \beta_k \kappa(\boldsymbol{x}_{\omega_k}, \boldsymbol{x}_t) + \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t) \\ &= \min_{\boldsymbol{\beta}}^{\top} \boldsymbol{\beta}\, \boldsymbol{K}_m\, \boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\, \boldsymbol{\kappa}(\boldsymbol{x}_t) + \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t). \end{aligned} \tag{8}$$

where $\boldsymbol{K}_m$ is the m-by-m Gram matrix of the m kernel functions, with entries $\kappa(\boldsymbol{x}_{\omega_i}, \boldsymbol{x}_{\omega_j})$. In this expression, $\boldsymbol{\kappa}(\cdot)$ is a column vector whose i-th entry is $\kappa(\boldsymbol{x}_{\omega_i}, \cdot)$; this is known in machine learning literature as the empirical kernel map [39], [44]. By taking the derivative of the above cost function with respect to $\boldsymbol{\beta}$, and setting it to zero, we get the optimal solution

$$\boldsymbol{\beta}_t = \boldsymbol{K}_m^{-1} \boldsymbol{\kappa}(\boldsymbol{x}_t). \tag{9}$$

By substituting this expression in (8), we get

$$\epsilon_t = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t) - \boldsymbol{\kappa}(\boldsymbol{x}_t)^\top \boldsymbol{K}_m^{-1}\, \boldsymbol{\kappa}(\boldsymbol{x}_t). \tag{10}$$

This expression defines the (squared) distance of $\kappa(\boldsymbol{x}_t, \cdot)$ to the subspace spanned by the m kernel functions.

Definition 1 (the distance criterion)**.** *Upon arrival of the $\boldsymbol{x}_t$ at instant t, we increment the model order by including the kernel function $\kappa(\boldsymbol{x}_t, \cdot)$ to the model if*

$$\epsilon_t > \nu, \tag{11}$$

for a given threshold ν; otherwise the kernel function is discarded and the model order remains unchanged.

This criterion is considered for instance in [26], [27] for sparse regression. From these papers (see also [23]), we give the following lemma.

Lemma 2. *Let $\kappa(\cdot,\cdot)$ be a reproducing kernel defined on a compact subspace $\mathcal{X}$. For any sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_\infty$, the model resulting from the distance criterion has a finite order.*

Sketch of proof: On the one hand, from the compactness of the subspace $\mathcal{X}$ and the continuity of $\kappa(\boldsymbol{x},\cdot)$, the resulting set of kernel functions, $\kappa(\boldsymbol{x}_1,\cdot), \kappa(\boldsymbol{x}_2,\cdot)\ldots,\kappa(\boldsymbol{x}_\infty,\cdot)$, is compact. Therefore, for any $\varepsilon > 0$, there exists a finite set of ℓ_2-balls of radius ε that *covers* these kernel functions. On the other hand, by construction, we have $\|\kappa(\boldsymbol{x}_{\omega_i},\cdot) - \kappa(\boldsymbol{x}_{\omega_j},\cdot)\|^2_{\mathcal{H}} > \nu$. By combining both statements, we get the desired result. □

Whilst this selection criterion seems unrelated to the eigen-decomposition problem, we study next the properties of the retained elements, and derive an upper bound on the approximation error of the principal functions.

3.2 Properties of the dictionary: the Gaussian kernel

In order to take advantage of the literature of dictionaries in sparse approximation [23], [45], [46], we consider the case of positive kernels with unit-norm[3], *i.e.*, $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_i) = 1$ for any $\boldsymbol{x}_i \in \mathcal{X}$. This is the case of the well-known Gaussian and exponential kernels. Next, we study some properties of the collection of m kernel functions, $\{\kappa(\boldsymbol{x}_{\omega_1},\cdot), \kappa(\boldsymbol{x}_{\omega_2},\cdot), \ldots, \kappa(\boldsymbol{x}_{\omega_m},\cdot)\}$, henceforth called dictionary.

A dictionary is c-coherent when the inner product between any pair of its elements does not exceed c, in absolute value, namely for any pair of elements

$$|\kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_j})| \leq c,$$

where the reproducing property $\langle \kappa(\boldsymbol{x}_{\omega_i},\cdot), \kappa(\boldsymbol{x}_{\omega_j},\cdot)\rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_j})$ is used.

Lemma 3. *The dictionary obtained by the distance criterion is $(1-\nu/2)$-coherent.*

Proof: By construction, we have $\|\kappa(\boldsymbol{x}_{\omega_i},\cdot) - \kappa(\boldsymbol{x}_{\omega_j},\cdot)\|^2_{\mathcal{H}} > \nu$, for any $i \neq j$. By expanding this norm, we get $2\kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_j}) < \kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_i}) + \kappa(\boldsymbol{x}_{\omega_j},\boldsymbol{x}_{\omega_j}) - \nu$. □

This result provides the relation between the threshold ν, the bandwidth σ of the Gaussian kernel, and the spread of the data, since the expression $|\kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_j})| \leq 1-\nu/2$ gives the relation

$$\|\boldsymbol{x}_{\omega_j} - \boldsymbol{x}_{\omega_j}\|^2 \geq -2\sigma^2 \ln(1-\nu/2).$$

Let $\boldsymbol{K}_m$ denotes the m-by-m Gram matrix associated with the dictionary, *i.e.*, with entries $\kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_j})$. The following theorem provides an upper and lower bounds on its eigenvalues. The following result is essentially due to [47], [48].

3. If the unit-norm is violated, such as for the quadratic kernel, one may substitute each $\kappa(\boldsymbol{x}_{\omega_i},\cdot)$ by its normalized counterpart $\kappa(\boldsymbol{x}_{\omega_i},\cdot)/\|\kappa(\boldsymbol{x}_{\omega_i},\cdot)\|$.

Theorem 4. *The eigenvalues of the dictionary Gram matrix satisfy the inequalities*

$$1-(m-1)(1-\nu/2) \leq \lambda_{\boldsymbol{K}_m} \leq 1+(m-1)(1-\nu/2).$$

Proof: The Geršgorin discs theorem, applied to matrix $\boldsymbol{K}_m$, states that the eigenvalues belong to the union of m discs. For $i = 1, 2, \ldots, m$, the i-th disc is centered on $\kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_i})$, a diagonal entry of $\boldsymbol{K}_m$, and has a radius of $\sum_{k=1,k\neq i}^{m} |\kappa(\boldsymbol{x}_{\omega_k},\boldsymbol{x}_{\omega_i})|$. Thus, all eigenvalues should satisfy

$$|\lambda_{\boldsymbol{K}_m} - \kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_i})| \leq \sum_{\substack{k=1\\k\neq i}}^{m} |\kappa(\boldsymbol{x}_{\omega_k},\boldsymbol{x}_{\omega_i})| \leq (m-1)(1-\nu/2),$$

for each $i = 1, 2, \ldots, m$, where the last inequality is due to Lemma 3. By taking each case, eigenvalue greater or lower than $\kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_i})$, we get the desired bounds. □

These bounds are sharp, in the sense that equality is attained for any dictionary of orthonormal kernel functions. By setting the threshold to $\nu = 2$, we get such dictionary, namely an incoherent dictionary since $\kappa(\boldsymbol{x}_{\omega_i},\boldsymbol{x}_{\omega_j}) = 0$ for any $i \neq j$. These results can be used for instance to give an upper bound on the corresponding condition number.

This theorem provides an upper bound on the eigenvalues captured by the Gram matrix of the dictionary. Still, we show next that such dictionary allows a good approximation of the exact, full-order, principal functions.

3.3 Error of approximating the principal functions

Let $\psi_r(\cdot)$ be the r-th principal function associated with the r-th largest eigenvalue λ_r, obtained from the whole t available data, say $\psi_r(\cdot) = \sum_{i=1}^{t} \alpha_{r,i}\, \kappa(\boldsymbol{x}_i,\cdot)$. The coefficients $\alpha_{r,i}$ correspond to the components of the r-th eigenvector of the Gram matrix $\boldsymbol{K}$, normalized such that $\sum_{i=1}^{t} \alpha_{r,i}^2 = 1/t\lambda_r$. In practice, the exact principal function is not known within an online setting. However, its approximation error with our approach and the subspace that we construct can be done with low error. The following theorem is essentially due to [26, Theorem 3.3] (with a slightly different proof).

Theorem 5. *The reduced-order model defined by the distance criterion can approximate the exact principal function $\psi_r(\cdot)$, with a (squared) approximation error upper bounded by*

$$\frac{\nu}{\lambda_r},$$

which is inversely proportional to its associated eigenvalue λ_r.

Proof: The norm of the residual approximation of the r-th principal function $\psi_r(\cdot)$ by the m kernel functions is defined as

$$\big\|(\boldsymbol{I}-\boldsymbol{P}_m)\psi_r(\cdot)\big\|^2_{\mathcal{H}} = \Big\| \sum_{i=1}^{t} \alpha_{r,i}(\boldsymbol{I}-\boldsymbol{P}_m)\kappa(\boldsymbol{x}_i,\cdot)\Big\|^2_{\mathcal{H}},$$

where the equality follows from the linearity of the operator. To provide an upper bound for this expression, we write

$$\begin{aligned}\left\|(\boldsymbol{I}-\boldsymbol{P}_m)\,\psi_r(\cdot)\right\|_{\mathcal{H}}^2 &= \Big\|\sum_{i=1}^{t}\alpha_{r,i}\,(\boldsymbol{I}-\boldsymbol{P}_m)\kappa(\boldsymbol{x}_i,\cdot)\Big\|_{\mathcal{H}}^2\\ &\leq \Big(\sum_{i=1}^{t}|\alpha_{r,i}|\,\big\|(\boldsymbol{I}-\boldsymbol{P}_m)\kappa(\boldsymbol{x}_i,\cdot)\big\|_{\mathcal{H}}\Big)^2\\ &\leq \sum_{i=1}^{t}|\alpha_{r,i}|^2\sum_{i=1}^{t}\big\|(\boldsymbol{I}-\boldsymbol{P}_m)\kappa(\boldsymbol{x}_i,\cdot)\big\|_{\mathcal{H}}^2\\ &\leq \Big(\sum_{i=1}^{t}\alpha_{r,i}^2\Big)\,t\,\nu\\ &= \frac{\nu}{\lambda_r},\end{aligned}$$

where the first inequality follows from the generalized triangular inequality, the second one is due to Cauchy-Schwarz inequality, the last inequality is due to the selection criterion, while the last equality follows from the normalization in the classical kernel-PCA. □

This result states that we can upper bound the error of approximating a principal function by a value inversely proportional to its associated eigenvalue. In other words, principal functions associated to high eigenvalues can be approximated with small errors.

4 Online kernel-PCA

Motivated by the theoretical results derived in the previous section, and more precisely the approximation bounds with Theorem 5, we consider a subspace approach to solve the kernelized Oja's updating rule, as studied next.

4.1 Learning the principal function

At instant t upon arrival of a new observation $\boldsymbol{x}_t$, the distance criterion is applied, leading to either case: the model-order is left unchanged or it is incremented. In both cases, the coefficients of the model are adapted appropriately, in a recursive scheme.

Case 1: $\epsilon < \nu$

In this case, the kernel function $\kappa(\boldsymbol{x}_t,\cdot)$ needs not to belong to the reduced-order model, since it can be approximated by its projection, $\kappa_p(\cdot)$. By substituting $\kappa(\boldsymbol{x}_t,\cdot)$ with $\kappa_p(\cdot)$ in expression (4), we get

$$\psi_{t+1}(\cdot) = \psi_t(\cdot) + \eta_t\,(y_t\kappa_p(\cdot) - y_t^2\,\psi_t(\cdot)).$$

Since $\kappa_p(\cdot) = \boldsymbol{\beta}_t^\top\boldsymbol{\kappa}(\cdot)$, where $\boldsymbol{\beta}_t$ is defined with equation (9), this leads to the following updating rule

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_t\,(y_t\boldsymbol{\beta}_t - y_t^2\,\boldsymbol{\alpha}_t). \tag{12}$$

This updating rule is essentially the kernelized version of Oja's updating rule (4), with the main difference being in the vector $\boldsymbol{\beta}_t$ which no longer is the sparse vector. In this expression, the output is computed from

$$y_t = \psi_t(\boldsymbol{x}_t) = \sum_{k=1}^{m}\alpha_{k,t}\kappa(\boldsymbol{x}_{\omega_k},\boldsymbol{x}_t),$$

which also corresponds to the output of the projected component[4]. It turns out that this is the reduced-order version (7) of the model defined by the kernelized Oja's rule, as in (5).

Case 2: $\epsilon > \nu$

In this case, the kernel function cannot be efficiently approximated by the model, and thus should be included to the model. This leads to a scheme similar to Oja's (4) with the model order incremented to $m+1$ and $\omega_{m+1} = t$.

The updating rule is given by

$$\boldsymbol{\alpha}_{t+1} = \begin{bmatrix}\boldsymbol{\alpha}_t\\ 0\end{bmatrix} + \eta_t\,y_t\left(\boldsymbol{\beta}_t - y_t\begin{bmatrix}\boldsymbol{\alpha}_t\\ 0\end{bmatrix}\right), \tag{13}$$

where $\boldsymbol{\alpha}_{t+1} = [\alpha_{1,t+1}\ \cdots\ \alpha_{m,t+1}\ \alpha_{m+1,t+1}]^\top$, and

$$\boldsymbol{\beta}_t = [0\ 0\ 0\ \cdots\ 0\ 0\ 1]^\top \tag{14}$$

is the sparse column vector of m zeros with the last entry set to 1, which we write $\boldsymbol{\beta}_t = [\boldsymbol{0}_m^\top\ 1]^\top$. In the expression above, the output is defined as

$$y_t = \psi_t(\boldsymbol{x}_t) = \sum_{k=1}^{m+1}\alpha_{k,t}\,\kappa(\boldsymbol{x}_{\omega_k},\boldsymbol{x}_t) = \boldsymbol{\alpha}_t^\top\boldsymbol{\kappa}(\boldsymbol{x}_t), \tag{15}$$

which is similar to the previous case.

The Gram matrix of the model and its inverse need to be updated only at the order-incrementation case, while they are required in the approximation case (Case 1) which often occurs. This can be seen with the $\boldsymbol{\beta}$'s in the projection expression. The Gram matrix is updated by adding a column and a row to the previous one, with

$$\boldsymbol{K}_{m+1} = \begin{bmatrix}\boldsymbol{K}_m & \boldsymbol{\kappa}(\boldsymbol{x}_t)\\ \boldsymbol{\kappa}(\boldsymbol{x}_t)^\top & \kappa(\boldsymbol{x}_t,\boldsymbol{x}_t)\end{bmatrix}.$$

One needs the inverse of this matrix in the computation of the approximation, by the evaluation of expression (9). For this, we invoke the well-known matrix identity

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix}^{-1} = \begin{bmatrix}\boldsymbol{A}^{-1} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} + \begin{bmatrix}-\boldsymbol{A}^{-1}\boldsymbol{B}\\ \boldsymbol{I}\end{bmatrix}\times (\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\begin{bmatrix}-\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I}\end{bmatrix}.$$

4. The output at any element corresponds to output of its projected component. This can be shown by decomposing $\kappa(\boldsymbol{x}_t,\cdot)$ into $\kappa_\perp(\cdot)$, a component orthogonal to the subspace of the m retained kernel functions, and $\kappa_p(\cdot)$, its projection onto the subspace. Then, the output is given as

$$y_t = \langle\psi_t(\cdot),\kappa(\boldsymbol{x}_t,\cdot)\rangle_{\mathcal{H}} = \langle\psi_t(\cdot),\kappa_p(\cdot)\rangle_{\mathcal{H}} + \langle\psi_t(\cdot),\kappa_\perp(\cdot)\rangle_{\mathcal{H}}$$

Since $\kappa_\perp(\cdot)$ is orthogonal to the subspace of $\psi_t(\cdot)$, *i.e.*, $\langle\psi_t(\cdot),\kappa_\perp(\cdot)\rangle_{\mathcal{H}} = 0$, we get

$$y_t = \langle\psi_t(\cdot),\kappa_p(\cdot)\rangle_{\mathcal{H}}.$$

Another way to see this, is to consider the output of the projected component (the right-hand-side of the above expression), namely $\boldsymbol{\alpha}_t^\top\boldsymbol{K}_m\boldsymbol{\beta}_t$, where $\boldsymbol{\beta}_t = \boldsymbol{K}_m^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_t)$ from (9), thus $\langle\psi_t(\cdot),\kappa_p(\cdot)\rangle_{\mathcal{H}} = \boldsymbol{\alpha}_t^\top\boldsymbol{\kappa}(\boldsymbol{x}_t)$.

By applying this equation to the above definition of $\boldsymbol{K}_{m+1}$, we get the following recursive expression

$$\boldsymbol{K}_{m+1}^{-1} = \begin{bmatrix} \boldsymbol{K}_m^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\epsilon_t^2} \begin{bmatrix} -\boldsymbol{\beta} \\ 1 \end{bmatrix} \begin{bmatrix} -\boldsymbol{\beta}^\top & 1 \end{bmatrix}. \qquad (16)$$

Therefore, we get the Gram matrix and its inverse without the need to recompute from scratch the whole matrices at each order-incrementation.

The resulting algorithm is summarized in Table 2. Next, we consider the problem estimating multiple principal functions.

4.2 Multiple principal functions

The proposed rule extracts online the first principal kernel function, with $\psi(\cdot) = \boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\cdot)$. The extension to multiple principal functions is straightforward, using the generalized Hebbian algorithm as proposed by Sanger for the linear principal component analysis [7], [8]. Let $\{\psi_{1,t}(\cdot), \psi_{2,t}(\cdot), \ldots, \psi_{r,t}(\cdot)\}$ denotes the collection of r principal functions to be determined, listed in descending order of eigenvalues. Then, the j-th principal function is given with

$$\psi_{j,t+1}(\cdot) = \psi_{j,t}(\cdot) + \eta_t \Big(y_{j,t}\, \kappa(\boldsymbol{x}_t, \cdot) - y_{j,t} \sum_{i=1}^{j} y_{i,t}\, \psi_{i,t}(\cdot) \Big), \qquad (17)$$

where $y_{j,t} = \psi_{j,t}(\boldsymbol{x}_t)$, for $j = 1, 2, \ldots, r$. This corresponds to a Gram-Schmidt orthonormalization process, since the above expression can be written as

$$\psi_{j,t+1}(\cdot) = \psi_{j,t}(\cdot) + \eta_t \Big(y_{j,t} \big(\kappa(\boldsymbol{x}_t, \cdot) - \sum_{i=1}^{j-1} y_{i,t}\, \psi_{i,t}(\cdot) \big) - y_{j,t}^2\, \psi_{j,t}(\cdot) \Big),$$

which is nothing but Oja's rule for $\psi_{j,t}(\cdot)$ with a modified input $\kappa(\boldsymbol{x}_t, \cdot) - \sum_{i=1}^{j-1} y_{i,t}\, \psi_{i,t}(\cdot)$, *i.e.*, the component of the data outside the span of previous principal functions. This condition restricts the principal functions to be orthogonal to each others. In matrix form, this leads to an expression of the form

$$\boldsymbol{\psi}_{t+1}(\cdot) = \boldsymbol{\psi}_t(\cdot) + \eta_t \Big(\boldsymbol{y}_t\, \kappa(\boldsymbol{x}_t, \cdot) - \mathrm{LT}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)\, \boldsymbol{\psi}_t(\cdot) \Big), \qquad (18)$$

where $\boldsymbol{\psi}_t(\cdot)$ is a *vector* of functions, $\boldsymbol{y}_t^\top = [y_{1,t} \;\; y_{2,t} \;\; \cdots \;\; y_{r,t}]$, and $\mathrm{LT}(\cdot)$ makes its argument lower triangular by setting to zero the entries above its diagonal. Now, we are in a position to derive the "multiple" version of the online kernel-PCA algorithm.

Upon arrival of any new observation $\boldsymbol{x}_t$ at instant t, two cases may arise depending if the distance criterion (11) is satisfied or not.

- If $\epsilon_t < \nu$, the model order remains unchanged, say $\psi_{j,t}(\cdot) = \sum_{k=1}^{m} \alpha_{j,k,t}\, \kappa(\boldsymbol{x}_{\omega_k}, \cdot)$ for $j = 1, 2, \ldots, r$. In (18), $\kappa(\boldsymbol{x}_t, \cdot)$ is substituted by its projection $\kappa_p(\cdot) = \boldsymbol{\beta}_t^\top \boldsymbol{\kappa}(\cdot)$. The resulting updating rule can be written as

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \eta_t \Big(\boldsymbol{y}_t\, \boldsymbol{\beta}_t - \mathrm{LT}(\boldsymbol{y}_t \boldsymbol{y}_t^\top) \boldsymbol{A}_t \Big), \qquad (19)$$

  where $\boldsymbol{A}_t = [\boldsymbol{\alpha}_{1,t} \;\; \boldsymbol{\alpha}_{2,t} \;\; \cdots \;\; \boldsymbol{\alpha}_{r,t}]$ is the m-by-r matrix whose j-th column corresponds to the coefficients of the j-th principal function with $\alpha_{j,1,t}, \alpha_{j,2,t}, \;\ldots\;, \alpha_{j,m,t}$.
- If $\epsilon_t > \nu$, the model order is incremented by adding $\kappa(\boldsymbol{x}_t, \cdot)$ to each of the r models with $\psi_{j,t}(\cdot) = \sum_{k=1}^{m+1} \alpha_{j,k,t}\, \kappa(\boldsymbol{x}_{\omega_k}, \cdot)$ for $j = 1, 2, \ldots, r$, where $\omega_{m+1} = t$. This leads to the matrix-form recursion

$$\boldsymbol{A}_{t+1} = \begin{bmatrix} \boldsymbol{A}_t \\ \boldsymbol{0}_r^\top \end{bmatrix} + \eta_t \left(\boldsymbol{y}_t \boldsymbol{\beta}_t - \mathrm{LT}(\boldsymbol{y}_t \boldsymbol{y}_t^\top) \begin{bmatrix} \boldsymbol{A}_t \\ \boldsymbol{0}_r^\top \end{bmatrix} \right),$$

  with $\boldsymbol{0}_r$ a column vector of r zeros, and $\boldsymbol{\beta}_t = [\boldsymbol{0}_m^\top \; 1]^\top$.

The resulting algorithm is similar to the one proposed in Table 2, with the following modifications: The coefficients are given by the matrix $\boldsymbol{A}_t$, and in the incremental step 4. it is given by $[\boldsymbol{A}_t^\top \;\; \boldsymbol{0}_r]^\top$. The output step 5. is substituted by $\boldsymbol{y}_t = \boldsymbol{A}_t^\top \boldsymbol{\kappa}(\boldsymbol{x}_t)$, and the updating step is given by expression (19). The centering expressions are identical to the ones presented in Appendix.

The choice of the appropriate number of retained components is not studied in this paper. One can still identify the optimal number of retained principal components r by studying the distribution of the eigenvalues, as often investigated in the conventional linear PCA algorithms. Moreover, the choice of r depends on the value of the threshold parameter ν. In fact, the latter determines the model order m, *i.e.*, the number of retained kernel functions in the span. Thus one should have r much smaller than m in order to define a relevant subspace of the span.

5 Discussions

Next, we study some issues in online learning with kernel-PCA: the choice of an appropriate stepsize, whether or not to center the data, and denoising with a pre-imag in a reduced-order model. Experimental results corroborate this theoretical analysis, as highlighted in next section.

5.1 Rate of convergence

To study the rate of convergence of multiple principal functions, we consider expression (17). It is obvious that principal functions associated with small eigenvalues cannot mature properly until after those of larger eigenvalues have. Moreover, since the effect of subtracting largest-variance directions from the data is to decrease the variance, this will slow learning of principal functions which mature later (see [49, page 52] for the convergence of the linear Hebbian learning algorithm). Since the generalized Hebbian algorithm is essentially a multi-Oja's rule, one often studies the convergence of a single Oja's updating rule.

In order to guarantee convergence, the authors of [50] derive implicit conditions on the stepsize parameter for

TABLE 2
Pseudocode and summary of the online kernel-PCA algorithm

Initialization	$m = 1, \omega_1 = 1, \boldsymbol{K}_1 = \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1), \boldsymbol{\beta}_1 = 1$
At each instant $t \geq 2$, upon acquisition of $\boldsymbol{x}_t$	
1. Compute $\boldsymbol{\kappa}(\boldsymbol{x}_t)$	$\boldsymbol{\kappa}(\boldsymbol{x}_t) = [\kappa(\boldsymbol{x}_{\omega_1}, \boldsymbol{x}_t) \ \cdots \ \kappa(\boldsymbol{x}_{\omega_m}, \boldsymbol{x}_t)]^\top$
2. Subspace representation of $\kappa(\boldsymbol{x}_t, \cdot)$	$\boldsymbol{\beta}_t = \boldsymbol{K}_m^{-1} \boldsymbol{\kappa}(\boldsymbol{x}_t).$
3. Compute (square) distance to subspace	$\epsilon_t^2 = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t) - \boldsymbol{\kappa}(\boldsymbol{x}_t)^\top \boldsymbol{\beta}_t$
4. IF the distance criterion is satisfied :	$\epsilon_t^2 \geq \nu$
Increment the model order	$m = m + 1, \ \omega_m = t, \ \boldsymbol{\alpha}_t = [\boldsymbol{\alpha}_t^\top \ 0]^\top$
Update the inverse of the Gram matrix,	$\boldsymbol{K}_m^{-1} = \begin{bmatrix} \boldsymbol{K}_{m-1}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\epsilon_t^2} \begin{bmatrix} -\boldsymbol{\beta}_t \\ 1 \end{bmatrix} \begin{bmatrix} -\boldsymbol{\beta}_t{}^\top & 1 \end{bmatrix}$
and the empirical kernel map $\boldsymbol{\kappa}(\boldsymbol{x}_t)$	$\boldsymbol{\kappa}(\boldsymbol{x}_t) = [\boldsymbol{\kappa}(\boldsymbol{x}_t)^\top \ \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t)]^\top$
Update subspace representation of $\kappa(\boldsymbol{x}_t, \cdot)$	$\boldsymbol{\beta}_t = [\boldsymbol{0}_{m-1}^\top \ 1]^\top$
5. Output $\psi_t(\boldsymbol{x}_t)$	$y_t = \boldsymbol{\alpha}_t^\top \boldsymbol{\kappa}(\boldsymbol{x}_t)$
6. Update the coefficients	$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_t \, y_t (\boldsymbol{\beta}_t - y_t \, \boldsymbol{\alpha}_t)$
Principal coordinate of any $\boldsymbol{x}$	$\psi(\boldsymbol{x}) = \boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\boldsymbol{x})$

iterative linear PCA. These results can be easily extended from the linear to the kernel-based PCA algorithms. For instance to get the expected convergence, the stepsize parameter should be smaller than the inverse of the largest eigenvalue. However, the eigenvalues are usually unknown. One needs therefore to estimate their values, and update them adaptively, as studied in [51] for iterative kernel-PCA. Yet, the computational effort for such calculations is quite expensive, making these techniques unadapted for online learning.

Back to the conventional Oja's rule, the convergence depends on a set of mathematical assumptions. Essentially, the stepsize parameter cannot be a constant, but has to decrease over time, such as $\eta_t = \eta_0/t$ where η_0 is a positive constant parameter. Such choice is common in the stochastic approximation literature, with slow convergence when η_0 is small and divergence for large values. In order to overcome these drawbacks and achieve convergence, a "search then converge" approach is investigated in [52], with

$$\eta_t = \frac{\eta_0}{1 + t/\tau}. \tag{20}$$

The tuning parameter τ determines the duration of the initial search phase, with $\eta_t \approx \eta_0$ (when $t \ll \tau$), before a converge phase where η_t decreases as η_0/t (when $t \gg \tau$). This widely used stepsize parameter is presented in [53] for iterative kernel-PCA, but abandoned in favor of computational-expensive eigenvalue-aware stepsize (as above).

In this paper, we consider the stepsize parameter as defined in (20). The choice of the tuning parameter τ depends on the application. For a stationary system, we consider large values of τ, leading to fast convergence. This is the case of the first series of experiments in next section. For non-stationary signals, a small value of τ allows more to track the evolution of the system with lower convergence. This is illustrated in the second application in experiments.

5.2 On centering the data in feature space

In conventional PCA and Oja's rule, data is assumed centered, *i.e.*, zero-mean. The kernel function maps the data into, almost always, an uncentered embedding. Actually, re-centering in the feature space can be done on the Gram matrix, without ever explicitly computing the map. For the kernel-PCA algorithm, one simply substitutes the Gram matrix $\boldsymbol{K}$ by the matrix

$$\boldsymbol{K} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{K} - \tfrac{1}{n}\boldsymbol{K}\mathbf{1}_n\mathbf{1}_n^\top + \tfrac{1}{n^2}\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{K}\mathbf{1}_n\mathbf{1}_n^\top, \tag{21}$$

where $\mathbf{1}_n$ is the n-entry 1's column vector. For the iterative kernel-PCA algorithm, the centering can be carried out iteratively as long as the number of samples is fixed, with the so-called semi-online approach [18].

For the proposed online kernel-PCA, the center is estimated in the tracked subspace, thus of the form $\overline{\boldsymbol{\beta}}_t^\top \boldsymbol{\kappa}(\boldsymbol{x})$, for a vector $\overline{\boldsymbol{\beta}}_t$ updated recursively (see Appendix). This leads to an expression similar to (21), with

$$\boldsymbol{K}_m - \mathbf{1}_m\overline{\boldsymbol{\beta}}_t^\top\boldsymbol{K}_m - \boldsymbol{K}_m\overline{\boldsymbol{\beta}}_t\mathbf{1}_m^\top + \mathbf{1}_m\overline{\boldsymbol{\beta}}_t^\top\boldsymbol{K}_m\overline{\boldsymbol{\beta}}_t\mathbf{1}_m^\top.$$

The resulting model output is give as $y_t = \boldsymbol{\alpha}_t^\top \boldsymbol{\kappa}^c(\boldsymbol{x}_t)$ for a single principal function, and

$$\boldsymbol{y}_t = \boldsymbol{A}_t^\top \boldsymbol{\kappa}^c(\boldsymbol{x}_t),$$

for multiple principal functions, where the initial vector $\boldsymbol{\kappa}(\boldsymbol{x}_t)$ in (15) is substituted with

$$\boldsymbol{\kappa}^c(\boldsymbol{x}_t) = \boldsymbol{\kappa}(\boldsymbol{x}_t) - \mathbf{1}_m\overline{\boldsymbol{\beta}}_t^\top\boldsymbol{\kappa}(\boldsymbol{x}_t) - \boldsymbol{K}_m\overline{\boldsymbol{\beta}}_t + \mathbf{1}_m\overline{\boldsymbol{\beta}}_t^\top\boldsymbol{K}_m\overline{\boldsymbol{\beta}}_t.$$

The derivation of these expressions is carried out in Appendix, where we derive an online centering within the tracked subspace. This leads to two variants of the proposed approach, an uncentered and a centered version.

In the uncentered setting, the resulting principal functions are linear combinations of the uncentered kernel functions. This corresponds to an online eigendecomposition of the non-central second moment matrix

of data in the feature space. In a more statistically appropriate principle, one considers zero-mean data, by simply centering at the origin. It remains an open question whether features should be extracted from a centered or an uncentered approach, even in the case of conventional (linear and batch) PCA. In a centered PCA, the variability about the center of the data is concerned, as opposed to the variability about the origin in its uncentered counterpart. Still, the relationships between both variants are strong, as studied in [54] and summarized here. The eigenvalues in the uncentered problem are interlaced with those in the centered one, and many eigenvectors are commonly similar. It is very often that the first eigenvector in the uncentered case is close to the direction that unites the origin and the center of the data. All these results, derived for linear PCA, can be easily extended to kernel-PCA and our online kernel-PCA approach. Experimental results highlights these results with the proposed algorithm.

5.3 Denoising scheme with a pre-image in the reduced-order model

The denoising in feature space is carried out in three stages.

First, the principal functions are estimated from a set of noisy observations, for instance with the method proposed in this paper. From n available data, let $\boldsymbol{A}_n$ denotes the resulting matrix of coefficients whose r columns define the r principal functions. It is assumed that these principal functions are noise-free, while noise is captured by the less relevant ones. Let $\boldsymbol{\psi}(\cdot) = [\psi_1(\cdot)\ \psi_2(\cdot)\ \cdots\ \psi_r(\cdot)]^\top$ denotes the *vector* of obtained functions (as defined for instance in (18)).

Second, to denoise any data $\boldsymbol{x}$ in the feature space, it is projected into the subspace spanned by the most relevant principal functions. Since these functions define a basis, we can write this projection

$$\begin{aligned}\boldsymbol{P}_{\boldsymbol{\psi}}(\kappa(\boldsymbol{x},\cdot)) &= \sum_{j=1}^{r}\langle\psi_j(\cdot),\kappa(\boldsymbol{x},\cdot)\rangle\,\psi_j(\cdot)\\ &= \sum_{k=1}^{m}\sum_{j=1}^{r}\psi_j(\boldsymbol{x})\,\alpha_{k,j}\,\kappa(\boldsymbol{x}_{\omega_k},\cdot)\\ &= \sum_{k=1}^{m}[\boldsymbol{A}_n\boldsymbol{y}]_k\,\kappa(\boldsymbol{x}_{\omega_k},\cdot),\end{aligned}$$

where $\boldsymbol{y} = \boldsymbol{A}_n^\top\boldsymbol{\kappa}^c(\boldsymbol{x})$. While the resulting projection is still centered, one may decenter by adding $\mu_n(\cdot)$, leading to

$$\boldsymbol{P}_{\boldsymbol{\psi}}(\kappa(\boldsymbol{x},\cdot)) + \mu_n(\cdot) = (\boldsymbol{A}_n\boldsymbol{y} + \overline{\boldsymbol{\beta}}_n)^\top\boldsymbol{\kappa}(\cdot). \qquad (22)$$

The resulting signature still lives in the feature space, often of infinite-dimension.

In the third stage, one needs to *pre-image* the pattern back to the input space, *i.e.*, find the pattern $\boldsymbol{x}^*$ in the input space whose image $\kappa(\boldsymbol{x}^*,\cdot)$ is as close as possible to the denoised signature (22). Several techniques have been derived to tackle this ill-posed problem. For a recent review, see [55]. While solving the pre-image problem is less tractable in full-order models, we show next how the reduced-order model provide tractable solutions.

In [56], we show that the pre-image can be roughly solved using a conformal-map approach, with only matrix inversion. Applied here for the signature (22), the pre-image solution is given by solving the linear system

$$\boldsymbol{X}_m^\top\boldsymbol{x}^* = \big(\boldsymbol{P}_m - \eta\boldsymbol{K}_m^{-1}\big)(\boldsymbol{A}_n\boldsymbol{y} + \overline{\boldsymbol{\beta}}_n)$$

where $\boldsymbol{X}_m = [\boldsymbol{x}_{\omega_1}\ \boldsymbol{x}_{\omega_2}\ \cdots\ \boldsymbol{x}_{\omega_m}]$, $\boldsymbol{P}_m = \boldsymbol{X}_m^\top\boldsymbol{X}_m$ is the matrix of inner products in the input space, and η a regularization parameter (set to 10 in experiments). Note that the matrix $\boldsymbol{K}_m$ is non-singular by construction, as stated in Theorem 4.

Another way to solve the pre-image problem, is to consider the minimization of

$$J(\boldsymbol{x}^*) = \|(\boldsymbol{A}_n\boldsymbol{y} + \overline{\boldsymbol{\beta}}_n)^\top\,\boldsymbol{\kappa}(\cdot) - \kappa(\boldsymbol{x}^*,\cdot)\|^2.$$

The gradient, associated with the gaussian kernel, is given as

$$\nabla_{\boldsymbol{x}}J(\boldsymbol{x}^*) = \frac{1}{\sigma^2}\sum_{k=1}^{m}[\boldsymbol{A}_n\boldsymbol{y}]_k\,(\boldsymbol{x}_{\omega_k}-\boldsymbol{x}^*)\exp(\|\boldsymbol{x}_{\omega_k}-\boldsymbol{x}^*\|^2/2\sigma^2).$$

Once again, the use of a reduced-order model provides a tractable expression with a summation over m entries, as opposed to the n-order expression for the full-order model. In [57], a gradient descent algorithm is proposed with a non-negativity constraint. It is essentially based the above gradient expression, with a stepsize weighted by the value of $\boldsymbol{x}^*$. This leads to a faster convergence towards zero-intensity pixels in image processing (see experimentations, where only a single step is applied).

6 Experiments

In this section, we illustrate the performance of the proposed approach, on synthetic and real datasets, and compare it to kernel-PCA and iterative kernel-PCA.

In order to measure the variance and its evolution, we consider the variance in the feature space. The variance explained by each principal function $\psi_{j,t}(\cdot)$ is defined at each instant t by

$$\mathrm{Var}_{\boldsymbol{x}}(\psi_{j,t}(\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{x}}\left(|\psi_{j,t}(\boldsymbol{x})|^2\right) - \left|\mathbb{E}_{\boldsymbol{x}}\left(\psi_{j,t}(\boldsymbol{x})\right)\right|^2, \quad (23)$$

where

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}}\big(\psi_{j,t}(\boldsymbol{x})\big) &= \mathbb{E}_{\boldsymbol{x}}\Big(\sum_{k=1}^{m}\alpha_{j,k,t}\,\kappa(\boldsymbol{x}_{\omega_k},\boldsymbol{x})\Big)\\ &= \sum_{k=1}^{m}\alpha_{j,k,t}\,\mathbb{E}_{\boldsymbol{x}}\big(\kappa(\boldsymbol{x}_{\omega_k},\boldsymbol{x})\big),\end{aligned}$$

and

$$\mathbb{E}_{\boldsymbol{x}}\big(|\psi_{j,t}(\boldsymbol{x})|^2\big) = \mathbb{E}_{\boldsymbol{x}}\left(\Big|\sum_{k=1}^{m}\alpha_{j,k,t}\,\kappa(\boldsymbol{x}_{\omega_k},\boldsymbol{x})\Big|^2\right).$$

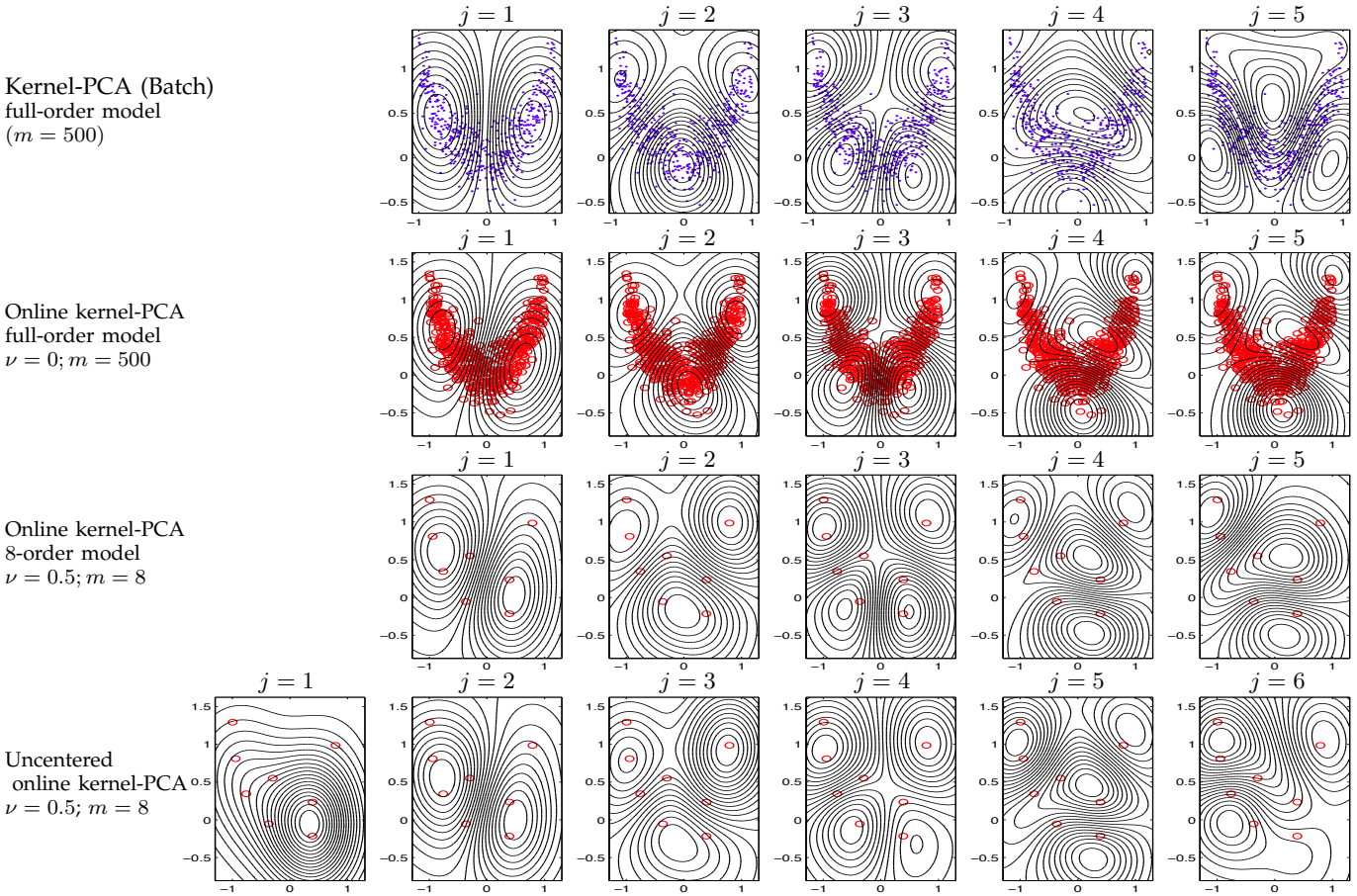

Fig. 1. Banana data with the contours of the first five principal functions with the Gaussian kernel, from $n = 500$ samples. The first three rows confronts classical kernel-PCA, iterative kernel-PCA (single pass), and online kernel-PCA. Results obtained with the uncentered version of our approach (fifth row) show that, besides the first principal function which captures the center of the data, all next five principal functions are quite similar to those obtained with the centered version (fourth row). Data points are given in the first row by blue dots ($\cdot$), while only elements contributing to the model are illustrated by red circles ($\circ$).

In practice, the expectation is estimated at each instant t over all available observations, including future samples $\boldsymbol{x}_{t'}$ for $t' > t$. The instant cumulative variance, explained by all r principal functions, is defined by $\sum_{j=1}^{r} \mathrm{Var}_{\boldsymbol{x}}(\psi_{j,t}(\boldsymbol{x}))$.

6.1 Banana-shaped 2-D distribution

In this first series of experiments, we compared the proposed approach to the classical kernel-PCA algorithm and the iterative kernel-PCA. A set of $n = 500$ two-dimensional data points in a banana-shaped distribution was used, as illustrated in Figure 1. The Gaussian kernel, often put forward as a *universal* kernel, was considered. The tuning parameters, which depend on the shaper and scale of the data distribution, were (naively) set to $\sigma = 0.5$ for the bandwidth of the Gaussian kernel, $\nu = 0.5$ for the selection threshold and $\eta_0 = 0.5$ for the asymptotic value of the stepsize parameter. The convergence parameter was set to $\tau = 10^5$ (a study of the convergence is conducted below).

With such value of the threshold, we get an 8-order model for each of the principal functions, namely 1.6% of the training data, as opposed to the full-order model (100%) for both the kernel-PCA and the iterative kernel-PCA. Essentially, the iterative kernel-PCA algorithm consists of applying our algorithm several times on the same data and setting $\nu = 0$. Similarities between the results obtained from these three algorithms are illustrated in Figure 1 (first, second and third row), with contour lines of the first five principal functions, $\psi_{j,t}(\cdot)$ for $j = 1, 2, \ldots, 5$ at $t = 500$. It is obvious that the proposed reduced-order model (with order $m = 8$ here) with the derived updating rule, captures the structure of the data, with basically the same performances as the (batch) kernel-PCA and the iterative kernel-PCA, both being full-ordered ($m = 500$).

These results were obtained from the *centered* versions of the considered algorithms, as often investigated in the literature. Still, one may also consider the uncentered version of our algorithm. As studied in Section 5.2 and illustrated in Figure 1 (last row), the first principal

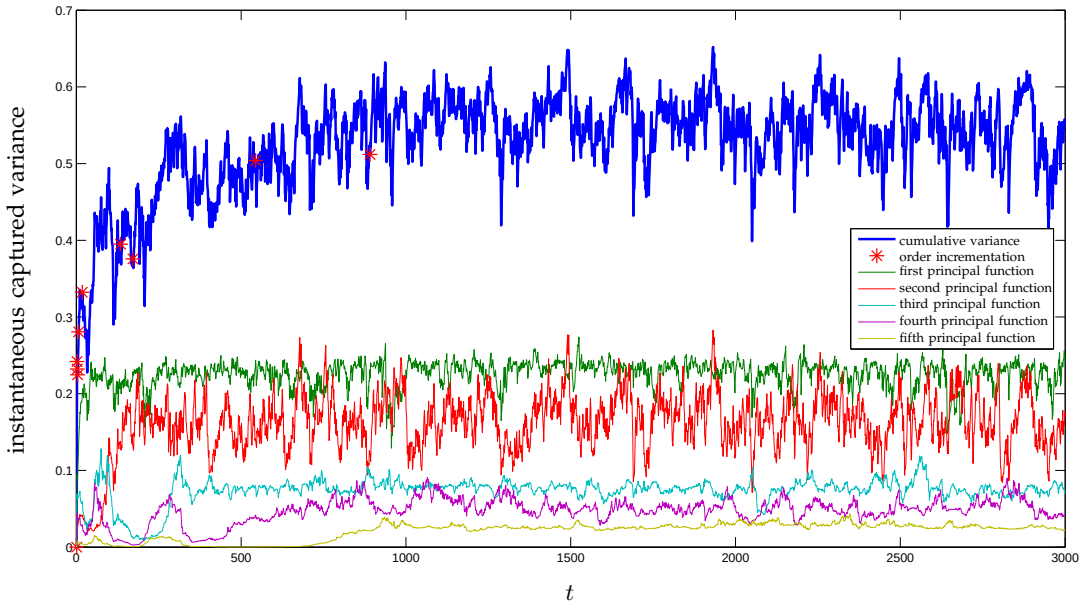

Fig. 2. Evolution of the variance explained by each principal function, and the cumulative variance explained by all the $r = 6$ principal functions. Instances of model incrementation are given by $\ast$.

function captures the variability about the center of the data. It is clear that all the other principal functions, $\psi_{j,t}$ for $j = 2, 3, \ldots, 5$ at $t = 500$, captures essentially the same structure as the centered version. Still, the last principal function ($j = 6$) did not mature enough at $t = 500$, and requires more data to converge to the optimal results. Next we study the convergence issue.

In order to study the convergence of the proposed algorithm (in its centered-version), we studied the instantaneous captured variance as defined by expression (23), where the expectations are estimated over all available samples. Moreover, we incremented the above set of data into $n = 3\,000$ samples. The selection criterion yields a 10-order model for each of the principal functions, namely 0.3% of the training data. Figure (2) illustrates the evolution of the variance explained by each of the five principal functions, as well as the cumulative variance. As expected, the relevance of each principal function is illustrated in terms of the explained variance. The convergence of the instant cumulative variance, explained by the five principal functions, depends on the choice of the convergence parameter τ. Figure 3 illustrates this influence, where small values of τ leads to slow convergence while large values correspond to a "search *without* convergence" strategy.

6.2 Image denoising

In this section, we consider an image processing application, with the MNIST database of handwritten digits [58]. This dataset consists of the handwritten digits "1", "2" and "3", given in 28-by-28 images with pixel-values normalized between 0 and 1. The dataset consists

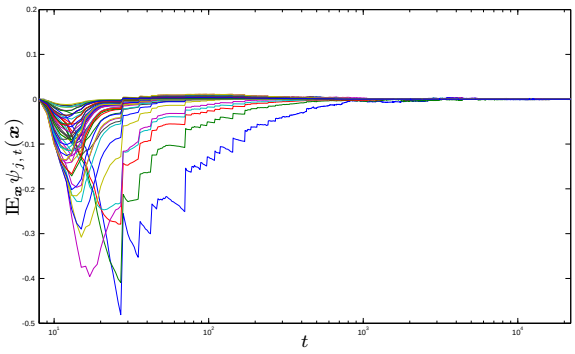

Fig. 5. Evolution of the mean of each principal function at each instant t, where $\mathbb{E}_{\boldsymbol{x}}\psi_{j,t}(\boldsymbol{x})$ is estimated with $\frac{1}{n}\sum_{\ell=1}^{n}\psi_{j,t}(\boldsymbol{x}_\ell)$ for $n = 22\,000$.

of $22\,008$ available images, treated as 784-dimensional vectors. All images were contaminated with a white Gaussian noise of zero-mean and variance 0.1. We studied a denoising scheme, where $n = 22\,000$ images were considered for learning the principal functions, and the remaining 8 images for denoising. We emphasize on the fact that the principal functions are determined from a collection of noisy images.

With such very largescale problem, the classical kernel-PCA is no longer tractable since it requires the diagonalization of a $22\,000$-by-$22\,000$ Gram matrix. An online scheme was considered here, where the images are presented one-by-one to the proposed (centered) online kernel-PCA algorithm. Each image was selected randomly from the set of available images, and is used

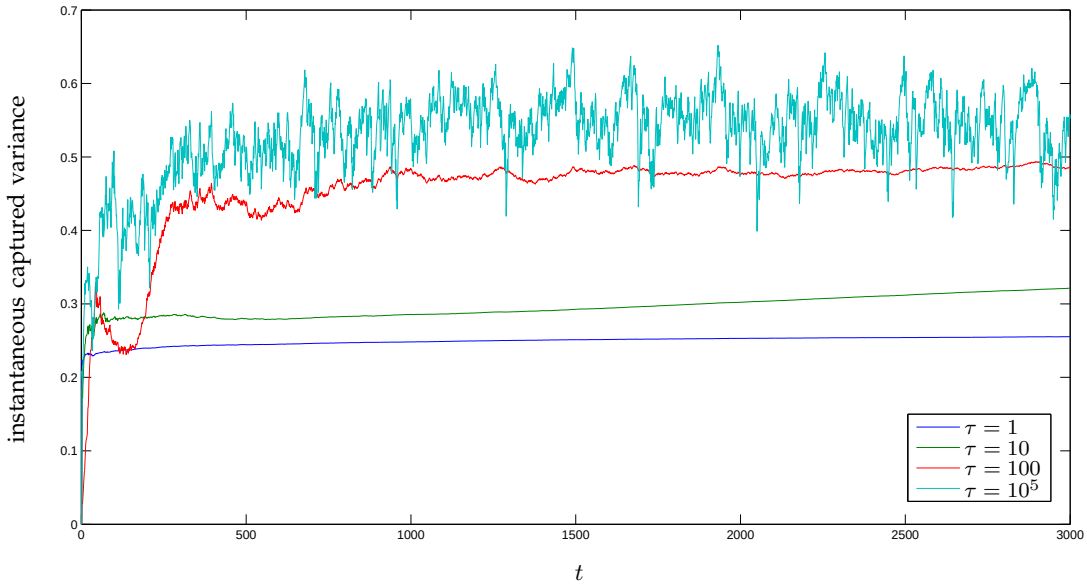

Fig. 3. Evolution of the variance explained by each principal function, for several convergence parameters.

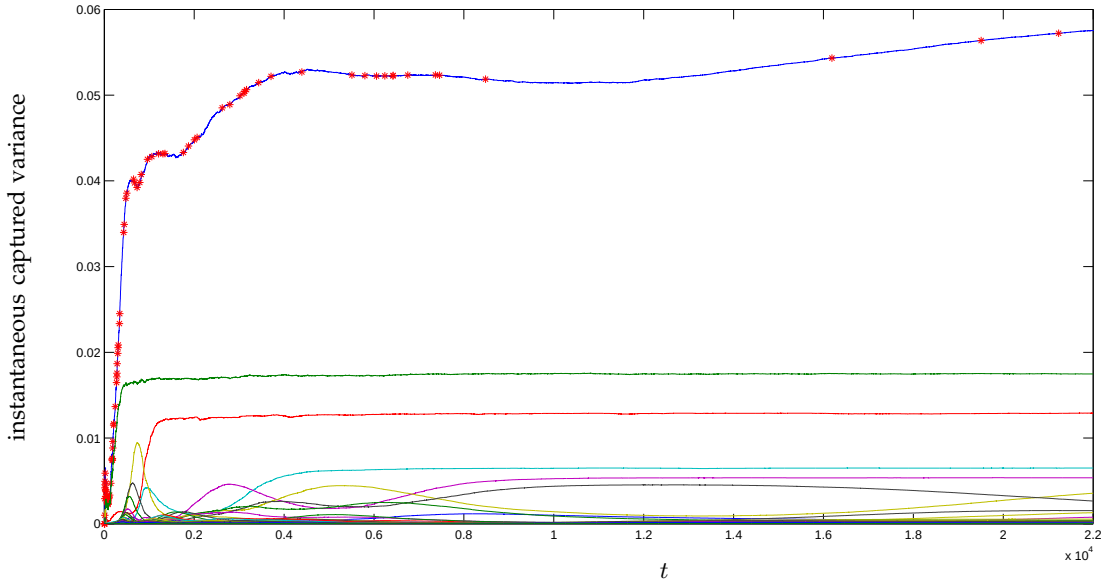

Fig. 4. Evolution of the variance explained by each principal function for the image denoising problem. The legend is the same as in Figure 2.

only once. The values of the tunable parameters were set as often recommended in the literature. The Gaussian kernel was used with its bandwidth set to $\sigma = 8.5$, as recommended in SVM classification task. As given in [53], the number of eigenvectors was set to 50, and the stepsize parameters were set to $\tau = 0.05n$ and $\eta = 1$ (see Section 5.1 for a discussion about the influence of these parameter). Preliminary experiments on the first 100 images were conducted in order to choose the value of the threshold in the selection criterion, which was set to $\nu = 0.55$. This leads to a model of final order $m = 90$, thus with only 0.4% of the available data.

Figure 4 gives the evolution of the captured variance, as defined in (23). As we considered the *centered* version of the online kernel-PCA, it turns our that the second term in right-hand-side converges to zero. This is illustrated in Figure 5 where, at every instant t, the principal functions were updated with the mean estimated over the whole dataset, namely $\frac{1}{n}\sum_{\ell=1}^{n}\psi_{j,t}(\boldsymbol{x}_\ell)$ for $n = 22\,000$. The relevance of the resulting principal functions is studied in a denoising scheme, as illustrated in Figure 6. As shown using two different pre-image

techniques (see Section 5.3), the resulting images are *clean*, even though the principal functions were trained from noisy images. With the iterative gradient technique, the stepsize was set to 100, and weighted by the value of $\boldsymbol{x}^*$, a trick that allows better convergence towards zero-intensity pixels. This leads to faster convergence, with the denoised results obtained from only one single gradient-descent step.

7 Conclusion

In this paper, we proposed an online kernel principal component algorithm. To this end, we studied the adaptation of Oja's rule to kernel machines, by proposing an appropriate rule to control the model order. We gave theoretical results on errors of using such reduced-order model. We derived a recursive algorithm for computing a principal function, and studied the case of having multiple functions. In the light of the considered model, we studied convergence rate, centering in feature space and denoising by pre-imaging. Experimental results show the relevance of these functions, on synthetic and real datasets. This work lay the ground for interesting future research questions. An adaptive stepsize is still an open question, and techniques such as Polyak-Ruppert averaging are investigated.

Acknowledgment

The author would like to thank Cédric Richard for the helpful discussions.

References

[1] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1986.
[2] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for modifying matrix factorizations," *Mathematics of Computation*, vol. 28, pp. 505–535, Apr. 1974.
[3] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, pp. 111–129, 1978.
[4] P. Hall, D. Marshall, and R. Martin, "Merging and splitting eigenspace models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 9, pp. 1042–1049, 2000.
[5] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math. Biology*, vol. 15, pp. 267–273, 1982.
[6] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *Journal of Mathematical Analysis and Applications*, vol. 106, pp. 69–84, 1985.
[7] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, pp. 459–473, 1989.
[8] T. D. Sanger, "Two iterative algorithms for computing the singular value decomposition from input/output samples," in *Proc. Advances in Neural Information Processing Systems 6* (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), pp. 144–151, 1993.
[9] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.
[10] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[11] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.
[12] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. Müller, "Fisher discriminant analysis with kernels," in *Advances in neural networks for signal processing* (Y. H. Hu, J. Larsen, E. Wilson, and S. Douglas, eds.), (San Mateo, CA, USA), pp. 41–48, Morgan Kaufmann, 1999.
[13] R. Rosipal and L. Trejo, "Kernel partial least squares regression in reproducing kernel hilbert space," *Journal of Machine Learning Research*, vol. 2, pp. 97–123, 2002.
[14] V. Guigue, A. Rakotomamonjy, and S. Canu, "Kernel basis pursuit," in *Proc. 16th European Conference on Machine Learning* (J. Gama, R. Camacho, P. Brazdil, A. Jorge, and L. Torgo, eds.), Lecture Notes in Computer Science, pp. 146–157, Springer, 2005.
[15] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
[16] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.
[17] K. Kim, M. Franz, and B. Schölkopf, "Kernel Hebbian algorithm for iterative kernel principal component analysis," Tech. Rep. 109, Max-Planck-Institut für Biologische Kybernetik, Tübingen, Germany, 06 2003.
[18] K. Kim, M. Franz, and B. Schölkopf, "Iterative kernel principal component analysis for image modeling," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1351–1366, 2005.
[19] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Processing*, vol. 52, Aug 2004.
[20] S.Smale and Y. Yao, "Online learning algorithms," *Found. Comput. Math.*, vol. 6, no. 2, pp. 145–170, 2006.
[21] S. V. Vishwanathan, N. N. Schraudolph, and A. J. Smola, "Step size adaptation in reproducing kernel hilbert space," *Journal of Machine Learning Research*, vol. 7, pp. 1107–1133, 2006.
[22] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, 2006.
[23] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Processing*, vol. 57, pp. 1058–1067, March 2009.
[24] G. Kimeldorf and G. Wahba, "Some results on tchebycheffian spline functions," *Journal of Mathematical Analysis and Applications*, vol. 33, pp. 82–95, 1971.
[25] B. Schölkopf, R. Herbrich, and R. Williamson, "A generalized representer theorem," Tech. Rep. NC2-TR-2000-81, NeuroCOLT, Royal Holloway College, University of London, UK, 2000.
[26] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares algorithm," *IEEE Trans. Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.
[27] L. Csató and M. Opper, "Sparse representation for gaussian process models," in *Advances in Neural Information Processing Systems 13* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), pp. 444–450, MIT Press, 2001.
[28] M. Ouimet and Y. Bengio, "Greedy spectral embedding," in *Proc. 10th International Workshop on Artificial Intelligence and Statistics* (R. G. Cowell and Z. Ghahramani, eds.), pp. 253–260, 2005.
[29] M. E. Tipping, "Sparse kernel principal component analysis," in *Advances in Neural Information Processing Systems 13* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), (Denver, CO, USA), pp. 633–639, MIT Press, 2001.
[30] A. Smola, O. Mangasarian, and B. Schölkopf, "Sparse kernel feature analysis," Tech. Rep. 99-04, University of Wisconsin, Data Mining Institute, Madison, 1999.
[31] Z. K. Gon, J. Feng, and C. Fyfe, "A comparison of sparse kernel principal component analysis methods.," in *Proc. Knowledge-Based Intelligent Engineering Systems and Allied Technologies* (R. J. Howlett and L. C. Jain, eds.), pp. 309–312, IEEE, 2000.
[32] G. P. McCabe, "Principal variables," *Technometrics*, vol. 26, pp. 137–144, May 1984.
[33] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational & Graphical Statistics*, vol. 15, pp. 265–286, June 2006.
[34] A. d'Aspremont, F. R. Bach, and L. E. Ghaoui, "Full regularization path for sparse principal component analysis," in *Proc. 24th international conference on Machine learning*, (New York, NY, USA), pp. 177–184, ACM, 2007.
[35] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Networks*, vol. 10, pp. 1000–1017, 1999.

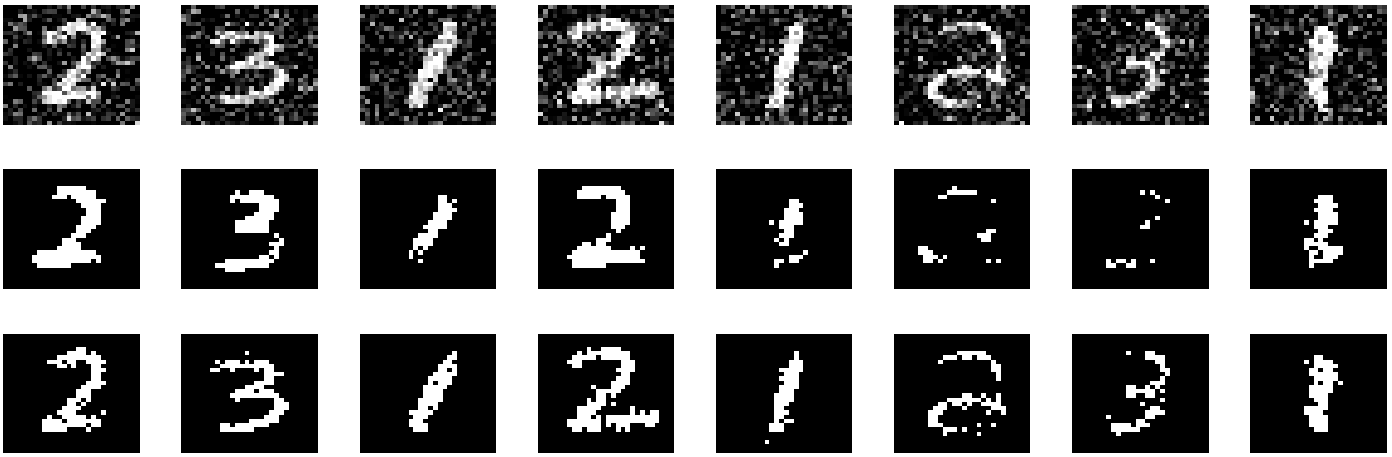

Fig. 6. The collection of 8 images, contaminated by noise (first row) and denoised by a our approach, using a matrix-inversion technique (second row) or an gradient descent technique (third row).

[36] L. Csató and M. Opper, "Sparse online gaussian processes," *Neural Computation*, vol. 14, pp. 641–668, 2002.

[37] M. Seeger, *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, Institute of Adaptive and Neural Computation, University of Edinburgh, Scotland, 2003.

[38] C. Burges, "Simplified support vector decision rules," in *Proc. 13th International Conference on Machine Learning*, pp. 71–77, 1996.

[39] M. Wu, B. Schölkopf, and G. Bakır, "A direct method for building sparse kernel learning algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 603–624, 2006.

[40] S. Agarwal, V. V. Saradhi, and H. Karnick, "Kernel-based online machine learning and support vector reduction," *Neurocomputing*, vol. 71, no. 7-9, pp. 1230–1237, 2008.

[41] T. Downs, K. E. Gates, and A. Masters, "Exact simplification of support vector solutions," *Journal of Machine Learning Research*, vol. 2, pp. 293–297, 2001.

[42] E. Parrado-Hernández, I. Mora-Jiménez, J. Arenas-García, A. R. Figueiras-Vidal, and A. Navia-Vázquez, "Growing support vector classifiers with controlled complexity," *Pattern Recognition*, vol. 36, no. 7, pp. 1479–1488, 2003.

[43] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *J. Mach. Learn. Res.*, vol. 7, pp. 1493–1515, 2006.

[44] B. Schölkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA, USA: MIT Press, 2002.

[45] J. A. Tropp, A. C. Gilbert, S. Muthukrishnan, and M. Strauss, "Improved sparse approximation over quasi-incoherent dictionaries," in *International Conf. on Image Processing*, vol. 1, pp. 37–40, 2003.

[46] A. C. Gilbert, S. Muthukrishnan, and M. J. Strauss, "Approximation of functions over redundant dictionaries using coherence," in *Proc. 14th ACM-SIAM symposium on Discrete algorithms*, (Philadelphia, PA), pp. 243–252, SIAM, 2003.

[47] J. A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Trans. Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[48] P. Honeine, C. Richard, and J. C. M. Bermudez, "On-line nonlinear sparse approximation of functions," in *Proc. IEEE International Symposium on Information Theory*, (Nice), pp. 956–960, June 2007.

[49] T. D. Sanger, "Optimal unsupervised learning in feedforward neural networks," tech. rep., MIT, Cambridge, MA, USA, 1989.

[50] L.-H. Chen and S. Chang, "An adaptive learning algorithm for principal component analysis," *IEEE Trans. Neural Networks*, vol. 6, pp. 1255 –1263, sep 1995.

[51] N. N. Schraudolph, S. Günter, and S. V. N. Vishwanathan, "Fast iterative kernel pca," in *Advances in Neural Information Processing Systems*, MIT Press, 2007.

[52] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Proc. IEEE Workshop on Neural Networks for Signal Processing*, (Piscataway, NJ), IEEE, 1992.

[53] S. Günter, N. N. Schraudolph, and S. V. N. Vishwanathan, "Fast iterative kernel principal component analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1893–1918, December 2007.

[54] J. Cadima and I. Jolliffe, "On relationships between uncentred and column-centred principal component analysis,," *Pakistan Journal of Statistics*, vol. 25, no. 4, pp. 473–503, 2009.

[55] P. Honeine and C. Richard, "Preimage problem in kernel-based machine learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 77–88, 2011.

[56] P. Honeine and C. Richard, "Solving the pre-image problem in kernel machines: a direct method," in *Proc. 19th IEEE workshop on Machine Learning for Signal Processing*, (Grenoble), Sept. 2009.

[57] M. Kallas, P. Honeine, C. Richard, C. Francis, and H. Amoud, "Non-negative pre-image in machine learning for pattern recognition," in *Proc. 19th European Conference on Signal Processing*, (Barcelona, Spain), 29 Aug. - 2 Sept. 2011.

[58] Y. Lecun and C. Cortes, "The mnist database of handwritten digits," *http://yann.lecun.com/exdb/mnist/*, 1998.

PLACE PHOTO HERE

Paul Honeine (M'07) was born in Beirut, Lebanon, on October 2, 1977. He received the Dipl.-Ing. degree in mechanical engineering in 2002 and the M.Sc. degree in industrial control in 2003, both from the Faculty of Engineering, the Lebanese University, Lebanon. In 2007, he received the Ph.D. degree in Systems Optimisation and Security from the University of Technology of Troyes, France, and was a Postdoctoral Research associate with the Systems Modeling and Dependability Laboratory, from 2007 to 2008. Since September 2008, he has been an assistant Professor at the University of Technology of Troyes, France. His research interests include nonstationary signal analysis and classification, nonlinear signal processing, sparse representations, machine learning, and wireless sensor networks. He is the co-author (with C. Richard) of the 2009 Best Paper Award at the IEEE Workshop on Machine Learning for Signal Processing.