Short-term time series prediction using Hilbert space embeddings of autoregressive processes

Edgar A. Valencia[a], Mauricio A. Álvarez[b,*]

[a]*Department of Mathematics, Faculty of Basic Sciences, Universidad Tecnológica de Pereira, Colombia, 660003.*
[b]*Department of Computer Science, Faculty of Engineering, The University of Sheffield, UK, S1 4DP*

Abstract

Linear autoregressive models serve as basic representations of discrete time stochastic processes. Different attempts have been made to provide non-linear versions of the basic autoregressive process, including different versions based on kernel methods. Motivated by the powerful framework of Hilbert space embeddings of distributions, in this paper we apply this methodology for the kernel embedding of an autoregressive process of order p. By doing so, we provide a non-linear version of an autoregressive process, that shows increased performance over the linear model in highly complex time series. We use the method proposed for one-step ahead forecasting of different time-series, and compare its performance against other non-linear methods.

Keywords: Autoregressive process, Hilbert space embeddings, cross-covariance operator, time series forecasting

1. Introduction

Autoregressive processes are useful probabilistic models for discrete time random processes. The basic idea in an autoregressive process is that the random variable at time n, can be described as a linear combination of the p past random variables associated to the process, plus white Gaussian noise. The value of p determines the order of the autoregressive process [1].

*Corresponding author
Email addresses: evalencia@utp.edu.co (Edgar A. Valencia), mauricio.alvarez@sheffield.ac.uk (Mauricio A. Álvarez)

Different authors have proposed non-linear extensions of the above model including NARMAX (non-linear autoregressive moving average model with exogenous inputs) [2], and also including the use of more general non-linear regression methods for extending the classical autoregressive process to non-linear setups. Examples of non-linear regression methods used are neural networks [3, 4], Gaussian processes [5, 6], and kernel-based learning methods [7, 8].

Within the kernel methods literature, different versions for kernelizing an autoregressive process of order p have been proposed [9, 7]. In [9], the authors propose an AR process built over a feature space. The coefficients of the autoregressive model are estimated by minimizing the quadratic error between the feature map of the input at time n, and the prediction given by the linear combination of the last p mapped inputs. Predictions are presented only for finite dimensional feature mappings, for which the inverse mapping from a feature space to the input space is easily computed. In [7], the authors also propose an AR process built over a feature space, by this time, the coefficients of the autoregressive model are estimated by using Yule-Walker equations, where the correlations between random variables are replaced by inner products between the feature maps of those random variables. Predictions are obtained by solving a pre-image problem.

Our objective in this paper is to introduce a non-linear version of the autoregressive model of order p based on Hilbert space embeddings of joint probability distributions.

Hilbert space embeddings are a recent trend in kernel methods that map distributions into infinite-dimensional feature spaces using kernels, such that comparisons and manipulations of these distributions can be performed using standard feature space operations like inner products or projections [10]. Hilbert space embeddings have been successfully used as alternatives to traditional parametric probabilistic models like hidden Markov models [11] or linear dynamical systems [12]. They have also been used as non-parametric alternatives to statistical tests [13].

Motivated by this powerful framework, we develop a kernelized version of an autoregressive model by means of the Yule-Walker algorithm, and instead of computing correlations (as in the classical AR linear model) or inner products (as in [7]), we compute cross-covariance operators for pairs of random variables. For time-series prediction, one additionally needs to solve a pre-image problem [14], to map from the space of covariance operators to the original input space. We develop an algorithm that uses fixed point iterations for solving the pre-image problem. The performance of the proposed model is compared against the linear AR model, the kernel method proposed in [7], neural networks, and Gaussian processes, for one-step ahead forecasting in different time series.

The paper is organized as follows. In section 2, we briefly review Hilbert space embeddings methods. In section 3, we present the embedding of the AR model using cross-covariance operators, including parameter estimation, and solving the pre-image problem. In section 4, we present some related work. In section 5 we describe the experimental setup that includes four datasets, and in section 6, we show the results for one-step ahead prediction over the different datasets. Conclusions appear in section 7.

2. Review of Hilbert space embeddings

In this paper, we use upper-case letters to refer to random variables (for example, X,Y), and lower-case letters to refer to particular values that those random variables can take (for example, x,y). Upper-case bold letters are used to refer to matrices, and lower-case bold letters are used for vectors.

We briefly review the definitions of a reproducible kernel Hilbert space (RKHS), Hilbert space embeddings of distributions, and covariance operators, which are the key for developing Hilbert space embeddings of autoregressive processes.

2.1. Reproducing Kernel Hilbert Space

A reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with kernel $k(x,x')$, for $x,x' \in \mathcal{X}$, is a space of functions $g : \mathcal{X} \to \mathbb{R}$ that satisfy the following properties:

1. For all $x \in \mathcal{X}$, $k(x,\cdot) : \mathcal{X} \to \mathbb{R}$ belongs to $\mathcal{H}$.

2. $\langle g(\cdot), k(x,\cdot)\rangle_{\mathcal{H}} = g(x)$ and consequently $\langle k(x,\cdot), k(y,\cdot)\rangle_{\mathcal{H}} = k(x,y)$.

An alternative definition for a kernel function, which is usually used when designing algorithms, is given by $k(x,x') = \langle \phi(x), \phi(x')\rangle_{\mathcal{H}}$, where $\phi : \mathcal{X} \to \mathcal{H}$.

Kernel methods are widely popular in signal processing and machine learning, and there are several textbooks where they are described in detail [15, 16, 17].

2.2. Embedding distributions

Recently, the authors in [13] introduced a method for embedding probability distributions in a RKHS. Let $\mathcal{P}_X$ be the space of all probability distributions $\mathbb{P}(X)$ of the random variable $X \in \mathcal{X}$. In [13], the authors define the mapping from a probability distribution $\mathbb{P}(X) \in \mathcal{P}_X$ to a RKHS $\mathcal{H}$ using the mean map μ_X defined as

$$\mu_X = \mathbb{E}_X[k(X,\cdot)] = \mathbb{E}_X[\phi(X)].$$

The mean map μ_X satisfies $\langle \mu_X, \phi(\cdot)\rangle_{\mathcal{H}} = \mathbb{E}_X[\phi(X)]$. If the kernel $k(x,x')$ used for the embedding is *characteristic*, [1] then μ_X is injective.

Given an *i.i.d.* set of observations $\{x^l\}_{l=1}^m$ of the random variable X, an estimator for $\widehat{\mu}_X$ is given as

$$\widehat{\mu}_X = \frac{1}{m}\sum_{l=1}^{m} k(x^l,\cdot).$$

It can be shown that $\langle \widehat{\mu}_X, \phi(\cdot)\rangle_{\mathcal{H}} = \frac{1}{m}\sum_{l=1}^{m}\phi(x^l)$. The estimator $\widehat{\mu}_X$ converges to μ_X, in the norm of $\mathcal{H}$, at a rate of $O_p(m^{-1/2})$ (see [13] for details).

2.3. Cross-covariance operator

If $\mathcal{H}_1$ and $\mathcal{H}_2$ are RKHS with kernels $k(\cdot,\cdot)$ and $\ell(\cdot,\cdot)$, and feature maps ϕ and φ, respectively, the *uncentered cross-covariance operator* is defined as [18]

$$\mathcal{C}_{XY} = \mathbb{E}_{XY}[\phi(X)\otimes\varphi(Y)],$$

where $\otimes$ is the *tensor product*.[2] The cross-covariance operator $\mathcal{C}_{XY}$ can be seen as an element of a *tensor product reproducing kernel Hilbert space* (TP-RKHS), $\mathcal{H}_1 \otimes \mathcal{H}_2$.

[1]A characteristic kernel is a reproducing kernel for which $\mu_X(\mathbb{P}) = \mu_Y(\mathbb{Q}) \iff \mathbb{P}=\mathbb{Q}$, $\mathbb{P},\mathbb{Q} \in \mathcal{P}$, where $\mathcal{P}$ denotes the set of all Borel probability measures on a topological space $(M,\mathcal{A})$.

[2]Given $f,h \in \mathcal{H}_1$, and $g \in \mathcal{H}_2$, we define the tensor product $f\otimes g$ as an operator that maps h from $\mathcal{H}_1$ to $\mathcal{H}_2$ such that $(f\otimes g)h \to \langle h, f\rangle_{\mathcal{H}_1} g$.

Given two functions $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$ then

$$\begin{aligned}\langle f, \mathcal{C}_{XY} g \rangle_{\mathcal{H}_1} &= \langle f \otimes g, \mathcal{C}_{XY} \rangle_{\mathcal{H}_1 \otimes \mathcal{H}_2} \\ &= \mathbb{E}_{XY}\left[\langle f \otimes g, \phi(X) \otimes \varphi(Y) \rangle_{\mathcal{H}_1 \otimes \mathcal{H}_2} \right] \\ &= \mathbb{E}_{XY}\left[\langle f,\ \phi(X) \rangle_{\mathcal{H}_1} \langle g,\ \varphi(Y) \rangle_{\mathcal{H}_2} \right] \\ &= \mathbb{E}_{XY}\left[f(X) g(Y) \right],\end{aligned}$$

where $\phi(x) = k(x,\cdot)$, $\varphi(y) = l(y,\cdot)$, and $\mathbb{E}_{XY}[f(x)g(y)]$ is the covariance matrix (for details see [19]).

The operator $\mathcal{C}_{XY}$ allows the embedding of $\mathcal{P}_{XY}$, the set of joint distributions $\mathbb{P}(X,Y)$ over the random vector $(X,Y) \in \mathcal{X} \times \mathcal{Y}$, in the TP-RKHS $\mathcal{H}_1 \otimes \mathcal{H}_2$.

Given an *i.i.d* of set of pairs of observations $\mathcal{D}_{XY} = \{(x^1, y^1), (x^2, y^2), \cdots, (x^m, y^m)\}$, a cross-covariance estimator $\widehat{\mathbf{C}}_{XY}$ for $\mathcal{C}_{XY}$ is defined as:

$$\widehat{\mathbf{C}}_{XY} = \frac{1}{m} \sum_{l=1}^{m} \phi(x^l) \otimes \varphi(y^l) = \frac{1}{m} \mathbf{\Phi} \mathbf{\Upsilon}^\top, \tag{1}$$

where $\mathbf{\Phi} = (\phi(x^1), \phi(x^2), \ldots, \phi(x^m))$, and $\mathbf{\Upsilon} = (\varphi(y^1), \varphi(y^2), \ldots, \varphi(y^m))$ are design matrices [17].

3. Hilbert space embedding of an autoregressive process

In this section, we describe how the basic autoregressive model can be embedded in a TP-RKHS. We then provide an estimation method for the parameters of the embedded method, by means of the Yule-Walker equations. Finally, we describe a procedure for solving the pre-image problem for the kernel embedding of the autoregressive process. We solve the pre-image problem for forecasting in time-series.

3.1. Autoregressive models in TP-RKHS

Let $X_1, X_2, \cdots, X_n$ a stationary discrete time stochastic process. A p-order linear AR model (LAR) is defined by [1]

$$X_i = \lambda_1 X_{i-1} + \lambda_2 X_{i-2} + \cdots + \lambda_p X_{i-p} + \varepsilon_i = \sum_{j=1}^{p} \lambda_j X_{i-j} + \varepsilon_i, \tag{2}$$

for $i = p+1, p+2, \cdots, n$, where $\lambda_1, \lambda_2, \cdots, \lambda_p$ are the model parameters, and ε_i is white noise with $\mathbb{E}(\varepsilon_i) = 0$ and $\mathrm{var}(\varepsilon_i) = \sigma^2$. We use $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_p]^\top$.

The Yule-Walker equations are a set of linear of equations used to estimate the coefficients $\boldsymbol{\lambda}$. The basic idea is to define a set of p linear equations, where the unknowns are the p coefficients in $\boldsymbol{\lambda}$. Each linear equation in the Yule-Walker system is formed by computing the covariance between X_i, and X_{i-k} according to

$$\langle X_i, X_{i-k} \rangle = \sum_{j=1}^{p} \lambda_j \langle X_{i-j}, X_{i-k} \rangle + \langle \varepsilon_i, X_{i-k} \rangle,$$

for $k = 1, \ldots, p$. Assuming independence between ε_i, and X_{i-k}, the set of linear equations reduce to

$$\langle X_i, X_{i-k} \rangle = \sum_{j=1}^{p} \lambda_j \langle X_{i-j}, X_{i-k} \rangle, \tag{3}$$

for $k = 1, \ldots, p$. Given a set of observations for the discrete time random process, and a suitable estimator for the covariance terms like $\langle X_i, X_{i-k} \rangle$, it is possible to solve the set of equations for estimating $\boldsymbol{\lambda}$.

The authors in [7] propose a non-linear extension of the AR process in (3), by applying a non-linear transformation $\varphi : \mathcal{X} \to \mathcal{H}$ to the random variables X_i in the AR model,

$$\varphi(X_i) = \sum_{j=1}^{p} \alpha_j \varphi(X_{i-j}) + \varphi(\varepsilon_i). \tag{4}$$

Notice that we use a set of coefficients $\boldsymbol{\lambda}$ for the autoregressive model in $\mathcal{X}$, and a set of coefficients $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_p]^\top$ for the autoregressive model in $\mathcal{H}$. To estimate the parameters $\boldsymbol{\alpha}$ in the transformed space, the authors follow a procedure similar to the Yule-Walker equations, but instead of computing covariances between random variables like X_i, and X_{i-k}, they compute inner products between $\varphi(X_i)$, and $\varphi(X_{i-k})$. With the proper independence assumptions, the Yule-Walker system of equations in then given as [3]

$$\langle \varphi(X_i), \varphi(X_{i-k}) \rangle = \sum_{j=1}^{p} \alpha_j \langle \varphi(X_{i-j}), \varphi(X_{i-k}) \rangle, \tag{5}$$

[3]For ease of exposition, we have assumed that the transformed random variables $\varphi(X_i)$ have been substracted the mean of the transformed variable $\mu_\varphi = \mathbb{E}_{X_i}[\varphi(X_i)]$.

for $k = 1, \dots, p$. Inner products like the ones above can be replaced by kernel functions. This is usually known as the *kernel trick* [15, 16]. Given a set of observations for the discrete time random process $\{x_i\}_{i=1}^m$, the following set of equations can be used to compute $\boldsymbol{\alpha}$,

$$k(x_i, x_{i-k}) = \sum_{j=1}^{p} \alpha_j k(x_{i-j}, x_{i-k}), \tag{6}$$

for $k = 1, \dots, p$. Since the values for $k(x_i, x_{i-j})$, and $k(x_{i-j}, x_{i-k})$ are themselves random variables that depend on the values of the observations in a particular time series, and assuming that the discrete time random process is stationary, the authors in [7] propose the following set of equations to get an estimate for $\boldsymbol{\alpha}$

$$\mathbb{E}[k(x_i, x_{i-k})] = \sum_{j=1}^{p} \alpha_j \mathbb{E}[k(x_{i-j}, x_{i-k})], \tag{7}$$

for $k = 1, \dots, p$. Expectations are estimated over the set of available samples. In this paper, we refer to the method by [7] as the Kernel autoregressive model (KAM).

Our key contribution in this paper is that we embedd the autoregressive model in a TP-RKHS by mapping joint distributions like $\mathbb{P}(X_i, X_{i-k})$, and $\mathbb{P}(X_{i-j}, X_{i-k})$ to points in $\mathcal{H}_1 \otimes \mathcal{H}_2$. Embeddings are performed by using cross-covariance operators, instead of inner products.

Let us start with Equation (4). If we apply a tensor product with $\phi(X_{i-k})$, at both sides of Equation (4), and take expected values, we obtain

$$\begin{aligned} \mathbb{E}_{X_i, X_{i-k}}[\varphi(X_i) \otimes \phi(X_{i-k})] = \sum_{j=1}^{p} \alpha_j \mathbb{E}_{X_{i-j}, X_{i-k}}[\varphi(X_{i-j}) \otimes \phi(X_{i-k})] \\ + \mathbb{E}_{\varepsilon_i X_{i-k}}[\varphi(\varepsilon_i) \otimes \phi(X_{i-k})], \end{aligned} \tag{8}$$

for $k = 1, \cdots, p$. If we assume that $\phi(X_{i-k})$, and $\varphi(\varepsilon_i)$ are uncorrelated, then the expression above reduces to

$$\mathcal{C}_{X_i X_{i-k}} = \sum_{j=1}^{p} \alpha_j \mathcal{C}_{X_{i-j} X_{i-k}}, \tag{9}$$

where $\mathcal{C}_{X_i X_{i-k}}$, and $\mathcal{C}_{X_{i-j} X_{i-k}}$ are cross-covariance operators, defined as

$$\begin{aligned} \mathcal{C}_{X_i X_{i-k}} &= \mathbb{E}_{X_i, X_{i-k}}[\varphi(X_i) \otimes \phi(X_{i-k})] \\ \mathcal{C}_{X_{i-j} X_{i-k}} &= \mathbb{E}_{X_{i-j}, X_{i-k}}[\varphi(X_{i-j}) \otimes \phi(X_{i-k})]. \end{aligned}$$

In this paper, we refer to this method as the Kernel embedding method (KEM).

Table 1 summarizes the different types of autoregressive processes described above.

Model	Space	Elements in space
Linear autoregressive model	Input space	$X_i \in \mathcal{X}$
Kernel autoregressive model [7]	Reproducible Kernel Hilbert Space (RKHS)	$\varphi(X_i) \in \mathcal{H}$
Kernel embedding method	Tensor product RKHS	$\mathcal{C}_{X_iX_j} \in \mathcal{H}_1 \otimes \mathcal{H}_2$

Table 1: Summary of the different autoregressive processes described in this section

3.2. Parameter estimation for autoregressive models in TP-RKHS

In this section, we provide a method for estimating the parameters $\boldsymbol{\alpha}$ in the autoregressive model in Equation (9). For this, we use the estimator for the cross-covariance operators, as in Equation (1).

Let $\mathcal{D}_{X_iX_{i-j}} = \{(x_i^1, x_{i-j}^1), (x_i^2, x_{i-j}^2), \cdots, (x_i^m, x_{i-j}^m)\}$, for $j = 1, 2, \cdots, p$, be different sets of samples drawn *i.i.d* from the distributions $\mathbb{P}(X_i, X_{i-j})$. We denote by $\mathbf{\Phi}_i$ the design matrix built from the elements $\{\phi(x_i^l)\}_{l=1}^m$, and $\mathbf{\Upsilon}_{i-j}$ the design matrix built from the elements $\{\varphi(x_{i-j}^l)\}_{l=1}^m$,

$$\mathbf{\Phi}_i = (\phi(x_i^1), \phi(x_i^2), \cdots, \phi(x_i^m)),$$

$$\mathbf{\Upsilon}_{i-j} = (\varphi(x_{i-j}^1), \varphi(x_{i-j}^2), \ldots, \varphi(x_{i-j}^m)).$$

Estimators for the cross-covariance operators $\mathcal{C}_{X_iX_{i-k}}$ and $\mathcal{C}_{X_{i-j}X_{i-k}}$ are given as (see Equation (9) and reference [11])

$$\widehat{\mathbf{C}}_{X_iX_{i-k}} = \frac{1}{m}\sum_{l=1}^{m} \phi(x_i^l) \otimes \varphi(x_{i-k}^l) = \frac{1}{m}\mathbf{\Phi}_i \mathbf{\Upsilon}_{i-k}^\top \tag{10}$$

$$\widehat{\mathbf{C}}_{X_{i-j}X_{i-k}} = \frac{1}{m}\sum_{l=1}^{m} \varphi(x_{i-j}^l) \otimes \varphi(x_{i-k}^l) = \frac{1}{m}\mathbf{\Upsilon}_{i-j} \mathbf{\Upsilon}_{i-k}^\top. \tag{11}$$

Equation (9) can now be written approximately as

$$\mathbf{\Phi}_i \mathbf{\Upsilon}_{i-k}^\top = \sum_{j=1}^{p} \alpha_j \mathbf{\Upsilon}_{i-j} \mathbf{\Upsilon}_{i-k}^\top. \tag{12}$$

We pre-multiply Equation (12) by $\boldsymbol{\Upsilon}_{i-k}^{\top}$, and post-multiply by $\boldsymbol{\Phi}_i$, obtaining

$$\boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Phi}_i\boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Phi}_i = \sum_{j=1}^{p}\alpha_j\boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Upsilon}_{i-j}\boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Phi}_i. \tag{13}$$

Simplifying

$$\boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Phi}_i = \sum_{j=1}^{p}\alpha_j\boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Upsilon}_{i-j}. \tag{14}$$

We can write the expression above as

$$\mathbf{H}_{i-k,i} = \sum_{j=1}^{p}\alpha_j\mathbf{K}_{i-k,i-j}, \tag{15}$$

where $\mathbf{H}_{i-k,i} = \boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Phi}_i$, $\mathbf{K}_{i-k,i-j} = \boldsymbol{\Upsilon}_{i-k}^{\top}\boldsymbol{\Upsilon}_{i-j}$, and $k = 1,2,\ldots,p$. Notice that the entries for the matrix $\mathbf{H}_{i-k,i}$ are the inner products $\{\varphi(x_{i-k}^{r})^{\top}\phi(x_i^{s})\}_{r=1,s=1}^{m,m}$. These inner products can be computed using a kernel function $\{h(x_{i-k}^{r},x_i^{s})\}_{r=1,s=1}^{m,m}$. Likewise, entries of $\mathbf{K}_{i-k,i-j}$ are given by inner products $\{\varphi(x_{i-k}^{r})^{\top}\phi(x_{i-j}^{s})\}_{r=1,s=1}^{m,m}$, which again can be computed using a kernel function $\{k(x_{i-k}^{r},x_{i-j}^{s})\}_{r=1,s=1}^{m,m}$.

Given a time-series dataset and a value for p, the values of $\mathbf{H}_{i-k,i}$, and $\mathbf{K}_{i-k,i}$ depend on the values chosen for i, and m. Assuming that the discrete time random process is stationary, we can get an estimate for $\boldsymbol{\alpha}$ using the following set of equations

$$\mathbb{E}[\mathbf{H}_k] = \sum_{j=1}^{p}\alpha_j\mathbb{E}[\mathbf{K}_{k,j}], \tag{16}$$

for $k = 1,\ldots,p$. We have suppresed the subindex i from the equation above to keep the notation uncluttered. As in Equation in (7), expectations can be estimated over the set of available samples.

We can use the system of equations in (16) to estimate the parameters $\boldsymbol{\alpha}$. The system of equations is given as

$$\begin{bmatrix}\mathbf{H}_1\\ \mathbf{H}_2\\ \vdots\\ \mathbf{H}_p\end{bmatrix} = \begin{bmatrix}\mathbf{K}_{1,1} & \mathbf{K}_{1,2} & \cdots & \mathbf{K}_{1,p}\\ \mathbf{K}_{2,1} & \mathbf{K}_{2,2} & \cdots & \mathbf{K}_{2,p}\\ \vdots & \vdots & \vdots & \vdots\\ \mathbf{K}_{p,1} & \mathbf{K}_{p,2} & \cdots & \mathbf{K}_{p,p}\end{bmatrix}\begin{bmatrix}\alpha_1\mathbf{I}\\ \alpha_2\mathbf{I}\\ \vdots\\ \alpha_p\mathbf{I}\end{bmatrix} \tag{17}$$

where $\mathbf{I}$ is the identity matrix of dimension m. We can find an estimator for $\boldsymbol{\alpha}$ by solving

$$\widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{H} - \mathbf{K}\boldsymbol{\alpha}_m\|_2^2, \tag{18}$$

where $\mathbf{H} \in \mathbb{R}^{mp\times m}$ is a block-wise matrix with blocks given by $\{\mathbf{H}_k\}_{k=1}^{p}$; $\mathbf{K} \in \mathbb{R}^{mp\times mp}$ is a block-wise matrix with blocks given by $\{\mathbf{K}_{k,j}\}_{k=1,j=1}^{p,p}$; and $\boldsymbol{\alpha}_m \in \mathbb{R}^{mp\times m}$ is also a block-wise matrix with blocks given as $\{\alpha_k \mathbf{I}\}_{k=1}^{p}$. For convenience, we also define $\widehat{\mathbf{K}}_i \in \mathbb{R}^{mp\times m}$ as a block-wise matrix taken from $\mathbf{K}$, with blocks given by $\{\mathbf{K}_{k,i}\}_{k=1}^{p}$.

It can be shown that the optimization problem in (18) can be cast into a least-squares problem as

$$\widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{A}\boldsymbol{\alpha} - \mathbf{b}\|_2^2, \tag{19}$$

where $\mathbf{A} \in \mathbb{R}^{p\times p}$ with entries $\{\mathrm{tr}(\widehat{\mathbf{K}}_i^\top \widehat{\mathbf{K}}_j)\}_{i=1,j=1}^{p,p}$, and $\mathbf{b} \in \mathbb{R}^{p\times 1}$ with entries $\{\mathrm{tr}(\mathbf{H}^\top \widehat{\mathbf{K}}_i)\}_{i=1}^{p}$.

3.3. Solving the pre-image problem for forecasting in a time-series

We want to use the method above for forecasting a new value x_i^* using $\boldsymbol{\alpha}$, and the p previous values of the time series. For now on, our method allows us to make predictions in the feature space by means of

$$\tau_i^* = \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}), \tag{20}$$

where the values for $\{\alpha_j\}_{j=1}^{p}$ have been estimated as explained in section 3.2. We would like to map back the value of τ_i^* to the input space, to get the predicted x_i^*. In the kernel literature this problem is known as the *pre-image problem* [14], and it is an ill-posed problem due to the higher dimensionality of the feature space, meaning that the transformed point τ_i^* may not have a corresponding x_i^* such that $\varphi(x_i^*) = \tau_i^*$.

We apply a tensor product to both sides of expression (20), leading to

$$\tau_i^* \otimes \phi(x_i^*) = \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}) \otimes \phi(x_i^*). \tag{21}$$

In order to get an estimate for x_i^*, we can solve the following minimization problem in $\mathcal{H}_1 \otimes \mathcal{H}_2$

$$x_i^* = \arg\min_x f(x) = \arg\min_x \left\| \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}) \otimes \phi(x) - \varphi(x) \otimes \phi(x) \right\|_{\mathcal{H}_1\otimes\mathcal{H}_2}^2,$$

where we have defined

$$f(x) = \left\| \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}) \otimes \phi(x) - \varphi(x) \otimes \phi(x) \right\|^2_{\mathcal{H}_1 \otimes \mathcal{H}_2}. \tag{22}$$

Expression for $f(x)$ can also be written as

$$\begin{aligned} f(x) = &\left\langle \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}) \otimes \phi(x), \sum_{k=1}^{p} \alpha_k \varphi(x_{i-k}) \otimes \phi(x) \right\rangle_{\mathcal{H}_1 \otimes \mathcal{H}_2} \\ &- 2 \left\langle \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}) \otimes \phi(x), \varphi(x) \otimes \phi(x) \right\rangle_{\mathcal{H}_1 \otimes \mathcal{H}_2} \\ &+ \langle \varphi(x) \otimes \phi(x), \varphi(x) \otimes \phi(x) \rangle_{\mathcal{H}_1 \otimes \mathcal{H}_2}. \end{aligned} \tag{23}$$

By using the property $\langle u \otimes v, a \otimes b \rangle_{\mathcal{H}_1 \otimes \mathcal{H}_2} = \langle u \otimes a \rangle_{\mathcal{H}_1} \langle v \otimes b \rangle_{\mathcal{H}_2}$, we get

$$\begin{aligned} f(x) = &\left\langle \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}), \sum_{k=1}^{p} \alpha_k \varphi(x_{i-k}) \right\rangle_{\mathcal{H}_1} \langle \phi(x), \phi(x) \rangle_{\mathcal{H}_2} \\ &- 2 \left\langle \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}), \varphi(x) \right\rangle_{\mathcal{H}_1} \langle \phi(x), \phi(x) \rangle_{\mathcal{H}_2} \\ &+ \langle \varphi(x), \varphi(x) \rangle_{\mathcal{H}_1} \langle \phi(x), \phi(x) \rangle_{\mathcal{H}_2}. \end{aligned} \tag{24}$$

Noticing that $C = \left\langle \sum_{j=1}^{p} \alpha_j \varphi(x_{i-j}), \sum_{k=1}^{p} \alpha_k \varphi(x_{i-k}) \right\rangle_{\mathcal{H}_1}$ is a constant (it does not depend on x), and using kernels $k(x,x')$ of the form $g(\|x-x'\|^2)$, we can simplify expression (24) as follows

$$f(x) = Cg(0) - 2g(0) \sum_{j=1}^{p} \alpha_j k(x_{i-j}, x) + g^2(0). \tag{25}$$

Taking the derivative with respect to x, we get

$$\frac{df(x)}{dx} = -2g(0) \sum_{j=1}^{p} \alpha_j \frac{dk(x_{i-j}, x)}{dx}. \tag{26}$$

If we use an squared exponential (SE) kernel or a radial basis function (RBF) kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2} \right), \tag{27}$$

where ℓ^2 is known as the bandwidth, the expression (26) follows as

$$\frac{df(x)}{dx} = -\frac{2g(0)}{\ell^2} \sum_{j=1}^{p} \alpha_j k(x_{i-j}, x)(x_{i-j} - x). \tag{28}$$

Equating to zero, and solving for x, we get the following fixed-point equation

$$x_i^* = \frac{\sum_{j=1}^{p} \alpha_j k(x_{i-j}, x_i^*) x_{i-j}}{\sum_{k=1}^{p} \alpha_k k(x_{i-k}, x_i^*)}. \tag{29}$$

4. Related work

Short term time series prediction is a classic topic in machine learning and statistics [20, 21, 22].

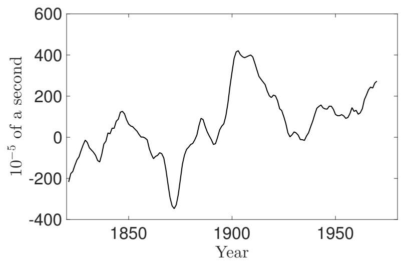
As we mentioned in the introduction, the authors in [9], and [7] introduced a kernelized version of an autoregressive process based on the kernel trick idea [15, 16]. In particular, the autoregressive model is built in a feature space, and the parameters of the model are estimated in two different ways, either by minimizing a quadratic error [9], or by means of the Yule-Walker algorithm [7]. In [9], the pre-image problem, this is, the problem of inverse transforming a point in the feature space, to the input space, is only solved for finite-dimensional feature spaces for which the inverse transformation can be readily be computed. In [7], the pre-image problem is solved by using a fixed-point algorithm similar to equation (29). When assuming a stationary kernel, this is $k(x,x') = k(x-x')$, the method in [7] turns out to be a particular example of the system in equation (15) for any particular values of r, and s in the kernel matrices $\mathbf{H}_{i-k,i}$, and $\mathbf{K}_{i-k,i-j}$.

Expression (10) follows closely Equation (5.15) in [23]. The expression in [23] is obtained as the Yule-Walker equations for a so called *autoregressive Hilbertian process of order p, ARH(p)*, that corresponds to an autoregressive process defined in a Hilbert space. In [23], the $\{\alpha_j\}_{j=1}^{p}$are bounded linear operators, in contrast to equation (10), where they correspond to scalar values. Estimation of α_j, and prediction are different though. The estimation for the bounded linear operators $\{\alpha_j\}_{j=1}^{p}$ is obtained by projecting the observations in a Hilbert space of finite dimension. Predictions are performed directly by applying the estimated operators over the input data.

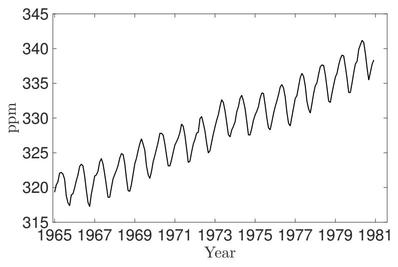
In [24], the author use kernel mean embeddings to provide one step ahead distribution prediction. In particular, distributions at any time t are represented by kernel mean maps. A mean map at time $t+1$ can be obtained as a mean map at time t, linearly transformed by a bounded linear operator. In fact, this corresponds to a ARH(1), where the functions in $\mathcal{H}$ correspond to kernel mean embeddings. The distribution at time $t+1$ in the input space is approximated by a weighted sum of historic input samples. The weigths in the approximation are computed from particular kernel expressions [24]. Our method considers models of order p, and our predictions are point estimates in contrast to [24]. Also, we use embeddings of joint probability distributions, $\mathbb{P}(X_i, X_{i-k})$ instead of embeddings of marginal distributions, $\mathbb{P}(X_i)$ (mean maps).

5. Experimental evaluation

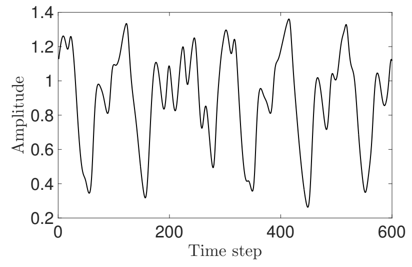
In this section, we provide details for the experimental evaluation performed in this paper. We describe tha datasets we use, and the procedure that we follow for validating the results.



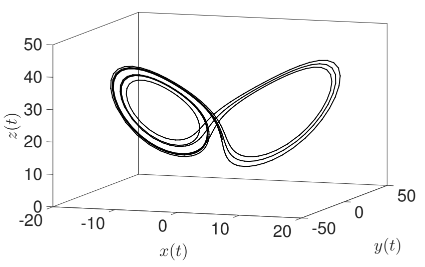
(a) Earthrot



(b) CO_2



(c) MG_{30}



(d) Lorenz

Figure 1: The four time-series used in this paper to compare the performance of the method proposed.

5.1. Datasets

We use four time-series to evaluate the performance of the different methods. The first two datasets belong to the Time Series Data Library (TSDL), and can be found in [25]. The last two datasets were generated by the authors.

– *Earthrot*. With the name *Earthrot*, we refer to the Annual changes in the earthÂ's rotation, day length (sec*10**-5) 1821-1970 dataset, available at [25]. Units are in 10^{-5} of a second. The time-series contains 150 samples. We use the first 130 samples for the experiments.

– *CO_2*. We use the dataset CO2 (ppm) mauna loa, 1965-1980 from the TSDL, which corresponds to monthly measures of CO_2 in parts per million from the Mauna Loa observatory. The time-series exhibit a periodic, and approximately linear behavior. The dataset contains 192 samples. For the experiments we use the first 150 samples.

– *MG_{30}*. The time-series MG_{30} refers to the time-series obtained from the Mackey-Glass non-linear time delay differential equation given as

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\tau)}{1+x(t-\tau)},$$

with $\tau = 30$ [7]. This time series exhibits chaotic dynamics. We generate a time-series of length 600. For the experiments, we use the first 400 samples.

– *Lorenz*. The Lorenz attractor refers to a set of three coupled ordinary differential equations given as

$$\begin{aligned}\frac{dx(t)}{dt} &= -ax + ay\\ \frac{dy(t)}{dt} &= -xz + rx - y\\ \frac{dz(t)}{dt} &= xy - bz,\end{aligned}$$

where a, r, and b are constants. For certain values a, r, and b, the system exhibits chaotic behavior. We set values for the parameters as $a = 10$, $r = 28$, and $b = 8/3$. For these values, the three-dimensional multi-variate time-series $(x(t), y(t), z(t))$

displays chaotic dynamics. We generate 500 samples per output dimension, and use the first 400 samples for the experiments. We perform prediction over the three time-series $x(t)$, $y(t)$, and $z(t)$, treating them as independent from each other.

Figure 1 shows the four datasets described above, and used for testing the methods.

5.2. Validation

The validation of the method proposed in this paper is done by performing one-step ahead prediction over each of the time series described above. For performing one step-ahead prediction, we use sliding frames of $w+1$ samples, where the first w samples are used for training, and the additional last sample is used for validation. The sliding frames are organized consecutively, with an overlap of w samples. The training data is used for setting the parameters of each of the models used for comparison, including the order of the autoregressive model. For the order of the model, we evaluate values of p from one to five. We compute the mean-squared error over the validating samples. We next describe the particular setup used for training in each of the models used in the experiments.

– *Linear AR model.* The coefficients $\boldsymbol{\lambda}$ for the linear AR model are estimated using the Yule-Waker equations. Within each frame of length w, we again use a sliding window of size $w/2+1$, where the first $w/2$ samples are used to compute $\boldsymbol{\lambda}$, and the last sample is used for performing one-step ahead prediction for different values of p. The sliding windows are organized consecutively with an overlap of $w/2$ samples. The results of the one-ahead step prediction withing the frame of length w, are used to select the value of p, which is selected as the value that ocurred more frequently offering the best prediction performance. Once the value for p has been selected, we compute again the values for $\boldsymbol{\lambda}$ using all the datapoints within w, and used this new $\boldsymbol{\lambda}$ for performing one-ahead step prediction over the time step $w+1$.

– *Kernel autoregressive model.* We implement the method proposed in [7]. To compute the expectations, we use sample means of the quantites of interest. For the kernel function, we use an SE kernel as in expression (27). The pre-image problem is solved as explained in [7], which has the same fixed-point solution as in expression (29). The values for ℓ, and p are chosen as follows: within the frame of length w, we generate sliding frames of size $w/2+1$. The sliding frames are organized consecutively with an overlap of $w/2$ samples. The first $w/2$ data points are used for estimating the values for $\boldsymbol{\alpha}$ by solving the system of equations in expression (7). We then use the data point at time step $w/2+1$ for selecting the best value for ℓ, and p, as the ones that on average, within the window of length w, yield the lowest error. We use a grid of values for ℓ by taking a grid of percentages, ℓ_p, of the median of the training data within the frame of size $w/2$. The percentages that we consider are 0.01, 0.01, 0.5, 1, 2, or 5 of the median of the training data within the frame of length $w/2$. Once we select the value for ℓ_p, we compute a new value for ℓ as the percentage ℓ_p of the median of the training data within the frame of size w. Having chosen ℓ, and p, we use all the training data of the frame of size w for finding a new set of coefficients $\boldsymbol{\alpha}$, and finally, provide a forecasting at time step $w+1$ by solving again a pre-image problem.

– *Kernel embedding method.* We implement the method described in section 3. We also use an SE kernel. The values for ℓ, and p used for one-step ahead forecasting for time step $w+1$ are computed as follows: within the frame of length w, we use an sliding window of length $w/2+1$. The sliding windows are set up consecutively with an overlap of $w/2$ samples. The first $w/2$ data points are used for estimating the coefficients $\boldsymbol{\alpha}$ by solving equation (19). For selecting ℓ and p, we follow a similar procedure to the one used for the kernel autoregressive model above: the sliding data point at time step $w/2+1$ is used to choose the values for ℓ, and p, that on average lead to the lowest prediction errors. Prediction at time step $w/2+1$ is performed by solving the pre-image problem in expression (29). In fact, we used percentages of the median of the training data, ℓ_p, in the windows of length $w/2$, in order to test different values for ℓ. The percentages that we used were 0.01, 0.01, 0.5, 1, 2, and 5. Once the best percentage of the median of the training data for ℓ, and the best value of the order of the model p have been chosen, we compute again the value for ℓ using the best value for ℓ_p and the training data in the whole frame of size w. We again compute $\boldsymbol{\alpha}$ using the training data in frame w, and forecast one-step ahead for the time setp $w+1$ solving the pre-image problem in expression (29).

– *Gaussian processes (GP).* We follow the model proposed in [5], in which the random variable of the process at time X_n can be described using

$$X_n = f(X_{n-1}, \dots, X_{n-p}) + \varepsilon, \tag{30}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, and $f(\mathbf{x})$ is assumed to follow a Gaussian process prior $f \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}))$, with covariance function $k(\mathbf{x}, \mathbf{x})$. For the covariance function, we use a SE kernel as in equation (27). The parameter of the covariance function ℓ, and the parameter σ for the likelihood model, are estimated by maximizing the log marginal likelihood using a scaled conjugate gradient procedure. We use the GPmat Toolbox[4] for all Gaussian processes related routines. For selecting the value of p, we use a sliding frame of length $w/2+1$, within the frame of length w. The sliding windows of length $w/2+1$ are established as in the other methods. A number of $w/2$ data points are used for learning the hyperparameters of the Gaussian process, and the data point at time step $w/2+1$ is used for cross-validating the value for p. The value for p is chosen as the one that on average (within the frame of size w) leads to the lowest errors. Once the value for p has been chosen, we use again the w samples for training a new GP. This new fitted GP is used for forecasting the data point at time step $w+1$.

– *Neural networks (NN).* We use a neural network with one hidden layer for learning a similar mapping as in equation (30). For choosing the number of neurons n_h of the hidden layer, and the value for p, we use a similar procedure as for the methods above: within the frame of length w, we generate sliding windows of length $w/2+1$, in a similar way as they were slid in the other approaches.

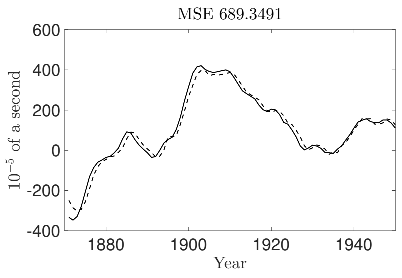
[4]Available at `https://github.com/SheffieldML/GPmat`

The $w/2$ first datapoints are used for fitting the weights of the neural network, and the data point at time step $w/2+1$ is used for choosing the value for n_h, and the value for p. These values are chosen as the ones that on average, within the frame of length w, lead to the lowest error. We allow the number of neurons in the hidden layer to be any of the following values: 5, 10, 15, 20, 25, or 30. For the the neural networks routines, we use the Neural Networks toolbox for MATLAB, with all the default settings, except for the number of neurons in the hidden layer.

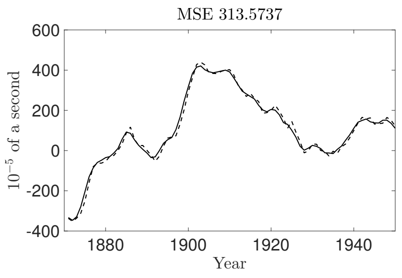
6. Results

We compare the performance of the different methods for short-term prediction over each of the time-series described in section 5. Figures 2, 3, 4, and 5 show the performance of the classical linear autoregressive model, the kernel autoregressive model, and the kernel embeddings of autoregressive model over the four time series described in Section 5. The mean squared error (MSE) for one step ahead prediction is shown as the title in each figure.

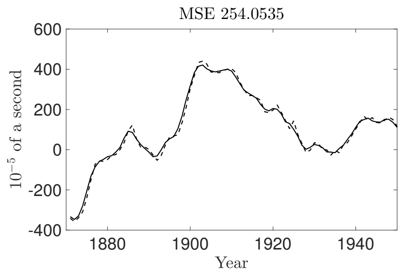
Figure 2 shows the one-step ahead prediction results for the time series Earthrot. For this example, we used sliding windows of length 51. The first 50 observations of each sliding window were used for training, and the forecast was perfomed for the time step 51-st of each sliding window. Since we used the first 130 samples from the original time series for the experiment, and a sliding window of 51 points, the MSE is computed over a total of 80 observation points. We notice that both kernel methods (figures 2(b) and 2(c)) are able to follow the original time series even from the first time steps, contrary to the linear model (figure 2(a)), where the prediction is far away from the time series. With respect to the 80 values of p that were chosen for each method, we computed a simple linear correlation coefficient between the series of values of p's for the linear method, and the two kernel approaches. As expected, there is a higher similarity between the values picked by the KAM, and the KEM, 0.5494, compared to -0.3349 for the correlation coefficient between the linear AR model and the KAM, and 0.2405 for the correlation coefficient between the linear AR model and KEM. A



(a) Earthrot using linear AR



(b) Earthrot using KAM

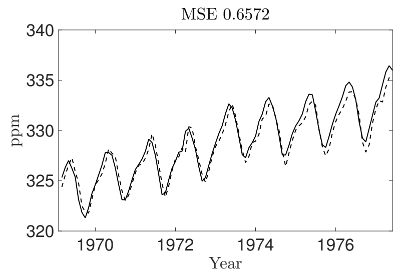


(c) Earthrot using KEM

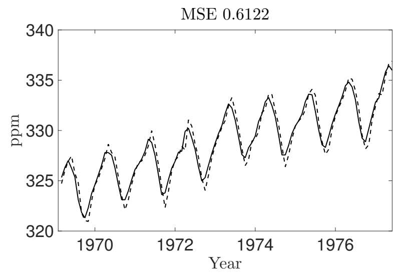
Figure 2: One-step ahead prediction over the dataset Earthrot given by the linear AR model, the method proposed by Kallas et. al. in [7], and the method based on kernel embeddings proposed in this paper. Solid lines are the test data, dashed lines are the predictions given by the methods. The title of each figure displays the mean squared error between the test data, and the predicted output.

further comparison between the values of p chosen by the kernel methods, show that they disagreed in 22 trials out of 80. With respect to the values of ℓ chosen by the two kernel methods, in only 4 out of the 80 trials, both methods chose different bandwidth values. The values for the MSE show that the method based on kernel embeddings offers the best performance when compared to the kernel autoregressive method, and the linear AR model.

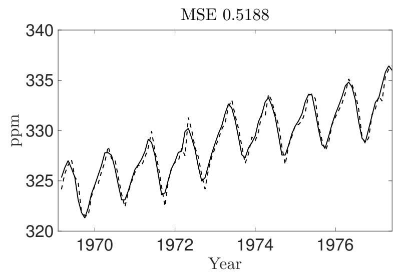
Figure 3 shows the results of one-step ahead forecasting for the linear AR model, the KAM, and the KEM. As in the previous example, we used sliding windows of length 51 samples, where the first 50 samples in each window are used for finding parameters of the models, and the last sample (number 51) is used to test the forecasting



(a) CO_2 using linear AR



(b) CO_2 using KAM



(c) CO_2 using KEM

Figure 3: One-step ahead prediction over the dataset CO_2, given by the linear AR model, the method proposed by Kallas et. al. in [7], and the method based on kernel embeddings proposed in this paper. Solid lines are the test data, dashed lines are the predictions given by the methods. The title of each figure displays the mean squared error between the test data, and the predicted output.

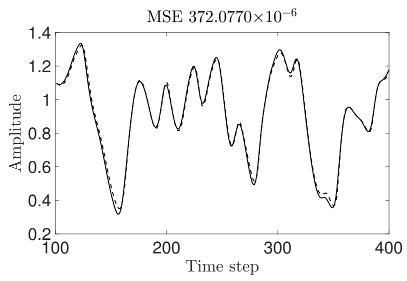
ability of the methods. From the CO_2 time series that is originally available, we used the first 150 samples to assess the prediction performance in several points of the time-series. Since we use window frames of 51 points, the MSE error for the prediction is computed over 100 samples of the time series.

It can be noticed how the KEM method is able to follow more closely the low and high peak values of the time series, when compared to the linear method, and KAM. This can be explained by the fact that the kernel embbeding method is able to take into account the particular structure in the time series, which for the KAM is lost when averaged. With respect to the values of p, the linear AR model chooses a value of $p = 2$, or $p = 5$, mostly. The KAM consistently worked better with $p = 2$, and the

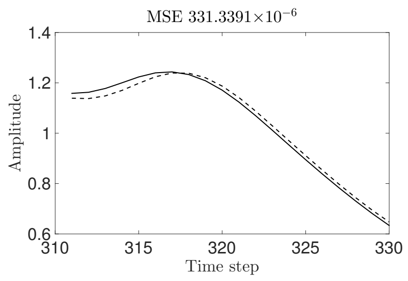
KEM with $p = 5$. The values chosen for ℓ in the kernel methods were equal 80% of the trails. The MSE values (appearing on the title of each figure) indicate that the KEM outperforms the linear AR method and the KAM.

Figure 4 shows the one-step ahead prediction for the Mackey-Glass chaotic time series. For this time series, we use sliding windows of length 101. The first 100 samples of the sliding window are used for training the models, and the sample 101-st is used for one-step ahead prediction. We perform the one-step ahead prediction over consecutive 300 samples, one at a time, and the MSE reported is the average over these 300 one-step ahead forecasting values. It can be noticed from figures 4(a), 4(c), and 4(e) that the methods based on kernels yield better prediction results than the linear method. Since it seems that qualitatively, the prediction performance for KAM and KEM is similar, we included additional figures where we zoom in a particular range where the difference in performace can be noticed. Figures 4(b), 4(d), and 4(f) show results for the MG_{30} time-series within a shorter time period, between time steps 311 and 330. With respect to the p values, the linear model favored a value of $p = 5$ (250 over the 300 trials). The KAM and the KEM predominantly used higher values of p: 70 for $p = 4$, and 163 for $p = 5$, for the KAM; and 43 for $p = 4$, and 257 for $p = 5$, for the KEM. In contrast to the experiments above, this time the kernel methods only selected the same value for ℓ in 76 cases out of 300. In the terms of the average MSE over the 300 trials, the experiment shows that both kernel mehods outperform the linear AR method. The MSE obtained by the KEM is lower than the one obtained by KAM.

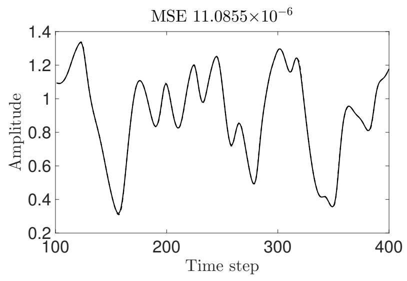
Figure 5 shows the prediction results over the Lorenz dataset. As explained before, prediction is performed over each component $(x(t), y(t), z(t))$ of the 3D time series, in an independent manner. For each of the three time series, we use sliding windows of length 101, where the forecasting is done over the last time step of each frame. The prediction performance is evaluated over 300 successive frames, all of them of length 101. With respect to the order p for the different models, the linear AR model picked $p = 2$ almost 22% of the time, and $p = 5$ almost 75% of the time. The KAM chose $p = 2$ almost 70% of all trials, and $p = 3$ almost 22% of all the repetitions. Finally, the KEM picked $p = 2$ almost 42% of the time, $p = 4$ almost 23% of the time, and $p = 5$ approximately 30% of the trials. As for the previous experiments, the kernel methods



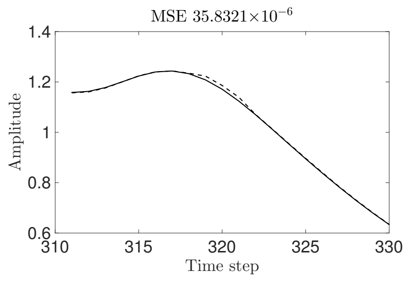
(a) MG$_{30}$ using linear AR



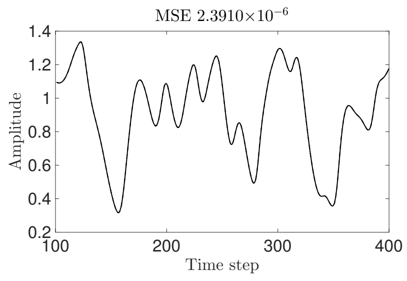
(b) MG$_{30}$ using linear AR



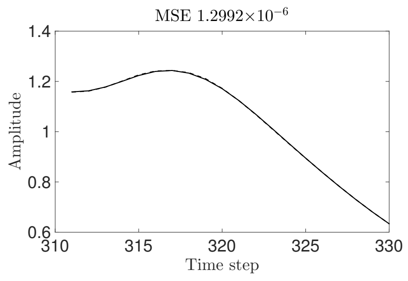
(c) MG$_{30}$ using KAM



(d) MG$_{30}$ using KAM

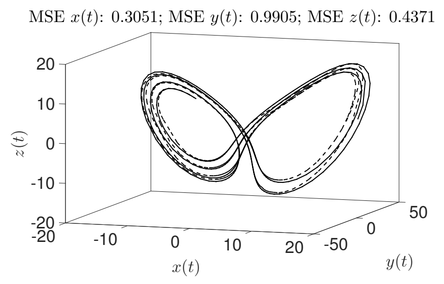


(e) MG$_{30}$ using KEM

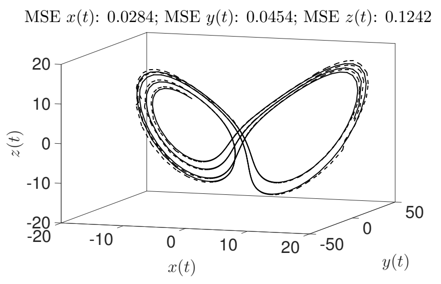


(f) MG$_{30}$ using KEM

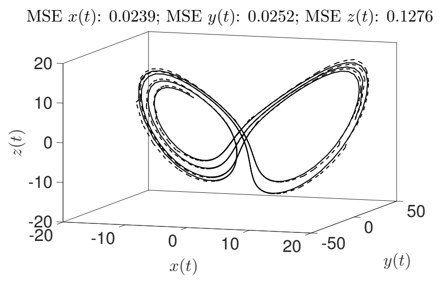
Figure 4: One-step ahead prediction over the MG$_{30}$ dataset given by the linear AR model, the method proposed by Kallas et. al.in [7], and the method based on kernel embeddings proposed in this paper. Solid lines are the test data, dashed lines are the predictions given by the methods. Figures 4(a), 4(c), and 4(e) show results for the MG$_{30}$ time-series. Figures 4(b), 4(d), and 4(f) show results for the MG$_{30}$ time-series within a shorter time period, between time steps 311 and 330. The title of each figure displays the mean squared error between the test data, and the predicted output.



(a) Lorenz using linear AR



(b) Lorenz using KAM



(c) Lorenz using KEM

Figure 5: One-step ahead prediction over the Lorenz dataset given by the linear AR model, the method proposed by Kallas et. al. (2013), and the method based on kernel embeddings proposed in this paper. Solid lines are the test data, dashed lines are the predictions given by the methods. The title of each figure displays the mean squared error between the test data, and the predicted output.

outperform the linear AR model. Although the prediction error of the KAM for $z(t)$ is lower than the prediction error for the KEM, on average, the KEM outperforms the KAM.

Table 2 shows a summary of the MSE obtained by the linear AR model, the kernel autoregressive model, and the kernel embedding method for the four datasets. It is clear from that table that the method that uses the kernel embeddings lead to better results, except of the component $z(t)$ of the Lorenz time series.

Table 3 shows the performance of neural networks, Gaussian processes, and the kernel autoregressive method compared to the performance of the kernel embeddings proposed in this paper. The value of w for all the time series was fixed to 50, and the

Table 2: Mean squared error for the test data and the predicted outputs, given by the linear autoregressive process (Linear AR), the kernel autoregressive model proposed by Kallas et al in [7] (KAM), and the kernel embeddings of autoregressive processes proposed in this paper (KEM). The values of the MSE for the MG_{30} should be multiplied by 10^{-6}.

Database	Linear AR	KAM	KEM
Earthrot	689.3491	313.5737	**254.0535**
CO_2	0.6572	0.6122	**0.5188**
MG_{30}	372.0770	11.0855	**2.3910**
Lorenz $x(t)$	0.3051	0.0284	**0.0239**
Lorenz $y(t)$	0.9905	0.0454	**0.0252**
Lorenz $z(t)$	0.4371	**0.1242**	0.1276

Table 3: Mean squared error for the test data and the predicted outputs, given by a neural network (NN), a Gaussian process regressor (GP), the kernel autoregressive model proposed by Kallas et al in [7] (KAM), and the kernel embeddings of autoregressive processes proposed in this paper (KEM). The values of the MSE for the MG_{30} should be multiplied by 10^{-6}.

Database	NN	GP	KAM	KEM
Earthrot	827.8469	570.1689	182.3038	**134.2564**
CO_2	0.6027	0.4631	0.4107	**0.3177**
MG_{30}	41.4519	2.0991	6.3064	**1.1052**
Lorenz $x(t)$	0.0595	0.0118	0.0088	**0.0037**
Lorenz $y(t)$	0.1114	0.0129	0.0146	**0.0038**
Lorenz $z(t)$	0.2041	0.0263	0.0211	**0.0140**

one-step ahead forecasting was performed for 80 time steps for Earthrot, 100 time steps for CO_2, and 300 for both MG_{30}, and Lorenz. The numerical optimization methods used for NN and GP are based on gradient-descent-like procedures, which heavily depend on a good parameter initialization to deliver sensible results. A bad parameter initialization often leads to poor prediction performance. In order to reduce the number of outliers for the prediction for NN and GP, we only computed the mean for those squared errors that were between quartiles 25-th and 75-th of all the squared errors computed for each time series. For a fair comparison, we also computed the MSE for the KAM, and the MSE for the KEM removing outliers, as explained before. We noticed from table 3 that the methods based on kernels, GP, KAM and KEM, yield better prediction performance than NN. Gaussian processes outperform KAM for the MG_{30} time series, and the component $y(t)$ of the Lorenz time series. The KEM method shows improved performance over all the other competing models.

To asses the statistical significance between the different methods, we applied the Diebold-Mariano test [26] to the forecasts given by the models. For Earthrot, we found that, with respect to our method, the null hypothesis of equal performances could be rejected for the NN and the GP. However, it could not be rejected for the KAM. For CO_2, we obtained a similar behavior: we could reject the hypothesis of equal performances between the NN and the KEM, and between the GP and the KEM. However, we could not reject the hypothesis of equal performance between the KAM and the KEM. For MG_{30}, we could reject the null hypothesis of equal performances between all the methods and our method. The same result was obtained for Lorenz $x(t)$, Lorenz $y(t)$, and Lorenz $z(t)$. For all the tests, we used a significance level of $\alpha = 0.05$. These results show that our method outperforms the KAM in four out of the six datasets analized. It also outperforms the NN and the GP for all the datasets that we studied.

7. Conclusions

In this paper, we have introduced kernel embeddings of joint probability distributions by means of an autoregressive process of order p placed over covariance operators. The solution to the model is done through a Yule-Walker system of equations for empirical estimates of the cross-covariance operators. Predictions in the input space are performed by solving a pre-image problem, for which a fixed-point algorithm is developed. Experimental results show that the method proposed here outperforms several non-linear versions of the autoregressive model, in the task of one-step ahead forecasting of time series. An important extension of this line work would be the formulation of a non-linear vector-valued autoregressive model, for which coefficients $\{\alpha_j\}_{j=1}^{p}$ would need to be considered as more general linear operators.

Acknowledgements

E. A. Valencia is being partly funded by University Tecnológica de Pereira. The authors would like to thank Colciencias, Colombia and British Council for funding under the project “Hilbert space embeddings of Autoregressive processes”. The authors would also like to thank Arthur Gretton, Zoltán Szabó and Kenji Fukumizu for their insightful comments and suggestions. We also appreciate the comments made by two anonymous reviewers, over a previous version of the manuscript. Part of this work was done while MAA was still Associate Professor at University Tecnológica de Pereira.

References

[1] K. S. Shanmugan, A. M. Breipohi, Random Signals: Detection, Estimation and Data Analysis, 1st Edition, Wiley, 1988.

[2] R. H. Schunway, D. S. Stoffer, Time series analysis an its aplications: with R examples, third edition Edition, Springer, 2011.

[3] O. Nelles, Nonlinear System Identification: from Classical approaches to Neural Networks and fuzzy models, first edition Edition, Springer, 2001.

[4] T. Y. Kim, K. J. Oh, C. Kim, J. D. Do, Artificial neural networks for non-stationary time series, Neurocomputing 61 (2004) 439 – 447, hybrid Neurocomputing: Selected Papers from the 2nd International Conference on Hybrid Intelligent Systems.

[5] J. Kocijan, A. Girard, B. Banko, R. Murray-Smith, Dynamic systems identification with, Gaussian processes, Mathematical and Computer modelling of Dynamical Systems 11(4) (2005) 411–424.

[6] J. Yan, K. Li, E. Bai, Z. Yang, A. Foley, Time series wind power forecasting based on variant Gaussian process and TLBO, Neurocomputing (2016) –In Press, Corrected Proof.

[7] M. Kallas, P. Honeine, C. Francis, H. Amoud, Kernel autoregressive models using Yule-Walker equations, Signal Processing 93 (2013) 3053–3061.

[8] W. He, Z. Wang, H. Jiang, Model optimizing and feature selecting for support vector regression in time series forecasting, Neurocomputing 72 (1-3) (2008) 600 – 611.

[9] R. Kumar, C. V. Jawahar, Kernel approach to autoregressive modeling, in: 13th National Conference on Comunications (NCC) Kanpur, India, 2007.

[10] L. Song, A. Gretton, K. Fukumizu, Embedding of conditional distribution, IEEE Signal Processing Magazine 30 (2013) 98–111.

[11] L. Song, S. M. Siddiqi, G. Gordon, A. Smola, Hilbert space embeddings of hidden Markov models, in: J. Fürnkranz, T. Joachims (Eds.), Proceedings of the 27th International Conference on Machine Learning (ICML-10), Omnipress, Haifa, Israel, 2010, pp. 991–998.

[12] L. Song, J. Huang, A. Smola, K. Fukumizu, Hilbert embeddings of conditional distributions with applications to dynamical systems, in: 26th Annual International Conference on Machine Learning Montreal-Canada, 2009, pp. 961–968.

[13] A. J. Smola, A. Gretton, L. Song, B. Schöllkopf, A Hilbert space embedding for distributions, in: 18th international conference on algorithmic learning theory: Springer-Verlag, Berlin, Germany, 2011, pp. 13–31.

[14] P. Honeine, C. Richard, Preimage problem in kernel-based machine learning, IEEE Signal Processing Magazine 28 (2011) 73–88.

[15] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Adaptive computation and machine learning, MIT Press, 2002.

[16] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.

[17] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[18] C. R. Baker, Joint measures and cross-covariance operators, American Mathematical Society 9 (186) 273–289.

[19] A. Gretton, O. Bousquet, A. Smola, B. Schölkopf, Algorithmic Learning Theory: 16th International Conference, ALT 2005, Singapore, October 8-11, 2005. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, Ch. Measuring Statistical Dependence with Hilbert-Schmidt Norms, pp. 63–77.

[20] G. Dudek, Neural networks for pattern-based short-term load forecasting: A comparative study, Neurocomputing 205 (2016) 64 – 74.

[21] P. Gupta, S. S. Batra, Jayadeva, Sparse short-term time series forecasting models via minimum model complexity, Neurocomputing 243 (2017) 1 – 11.

[22] R. Palivonaite, K. Lukoseviciute, M. Ragulskis, Short-term time series algebraic forecasting with mixed smoothing, Neurocomputing 171 (2016) 854–865.

[23] D. Bosq, Linear processes in function spaces: theory and applications, Springer, 2000.

[24] C. H. Lampert, Predicting the future behavior of a time-varying probability distribution, in: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, 2015, pp. 942–950.

[25] R. Hyndman, Time series data library, `http://data.is/TSDLdemo`.

[26] F. Diebold, R. Mariano, Comparing predictive accuracy, Journal of Business and Economic Statistics 13 (1995) 253–65.