Time Series Filtering, Smoothing and Learning using the Kernel Kalman Filter

Conference Paper · January 2005

DOI: 10.1109/IJCNN.2005.1556088 · Source: IEEE Xplore

2 authors:

Liva Ralaivola
Aix-Marseille Université
85 PUBLICATIONS **1,382** CITATIONS
SEE PROFILE

Florence d'Alché-Buc
Télécom ParisTech
114 PUBLICATIONS **1,321** CITATIONS
SEE PROFILE

Nonlinear Time Series Filtering, Smoothing and Learning using the Kernel Kalman Filter

Liva Ralaivola LIVA.RALAIVOLA@LIP6.FR
Laboratoire d'Informatique de Paris 6, 8, rue du Capitaine Scott, F-75015 Paris, France
Epigenomics Project, Genopole, 93, rue Henri Rochefort, F-91000 Evry, France

Florence d'Alché-Buc DALCHE@EPIGENOMIQUE.ORG
Epigenomics Project, Genopole, 93, rue Henri Rochefort, F-91000 Evry, France

Abstract

In this paper, we propose a new model, the Kernel Kalman Filter, to perform various nonlinear time series processing. This model is based on the use of Mercer kernel functions in the framework of the Kalman Filter or Linear Dynamical Systems. Thanks to the kernel trick, all the equations involved in our model to perform filtering, smoothing and learning tasks, only require matrix algebra calculus whilst providing the ability to model complex time series. In particular, it is possible to learn dynamics from some nonlinear noisy time series implementing an exact EM procedure. When predictions in the original input space are needed, an efficient and original preimage learning strategy is proposed.

1. Introduction

Time series modeling has become an appealing field for machine learning approaches over the last decade. Time series analysis has indeed connections to economy forecasting, biological sequences study, speech recognition, video processing and prediction of Web-user behavior. In front of a large amount of data which often take the form of non linear time series, there is a need of transversal and flexible tools that can be engineered easily. In this paper, we propose a kernel-based approach to time series modeling enabling the implementation of prediction, denoising and learning tasks. On the one hand, our method presents the advantage of keeping the framework of linear dynamical systems usable, and, on the other hand, the processing of non vectorial time series can be considered.

The paper is organized as follows. In section 2, we describe our Kernel Kalman Filter (KKF) model. Section 3 focuses on a few works which are closely related to ours, while section 4 reports empirical results achieved by our model on various times series processing tasks. Eventually, section 5 gives hints about future developments of KKF.

2. Kalman Filtering with Kernels

2.1. Kalman Filter Model

The Kalman filter is based on the idea that a dynamical process is *hidden* and can only be *observed* or *measured* through some time series. The dependency between two consecutive *states* of the process is assumed linear as well as the dependency between the *measurements* and the state process.

Assume a sequence $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ of T observation vectors in $\mathbb{R}^d$. The Kalman filter model (Kalman, 1960) assumes the following model for the series $\mathbf{x}_{1:T}$:

$$\mathbf{s}_{t+1} = A\mathbf{s}_t + \boldsymbol{\mu}_s + \boldsymbol{\nu}_s \quad (1a)$$
$$\mathbf{x}_t = C\mathbf{s}_t + \boldsymbol{\mu}_x + \boldsymbol{\nu}_x \quad (1b)$$

where $\boldsymbol{\nu}_s$ and $\boldsymbol{\nu}_x$ are zero-mean gaussian noise vectors of variances σ_s^2 and σ_x^2, respectively; if n is the dimension of $\mathbf{s}_t$, A is a $n \times n$ matrix, $\boldsymbol{\mu}_s$ an n-dimensional vector, C a $d \times n$ matrix and $\boldsymbol{\mu}_x$ a d-dimensional vector. The state process (1a) is intended to model the actual transition between two states of the studied process whereas the measurement process is (1b). It can be noticed that the studied series must have intrinsically linear dynamics to be modelled accurately by (1).

Two kinds of tasks are associated with the model (1). The first one encompasses *filtering* and *smoothing* and it deals with the *estimation* of the state vector $\mathbf{s}_t$ when the parameters $\boldsymbol{\theta} = \{A, \boldsymbol{\mu}_s, \sigma_s^2, \boldsymbol{\mu}_1, \sigma_1^2, C, \boldsymbol{\mu}_x, \sigma_x^2\}$ are known and a series $\mathbf{x}_{1:T}$ is observed. The second issue deals with the situation where the parameters $\boldsymbol{\theta}$ of the model have to be *learned*: this task is classically addressed using an *Expectation-Maximization* (EM) (Dempster et al., 1977) algorithm.

A set of *Kalman Filter* and *Kalman Smoother* equations is available to perform the estimation of the states. The filtering equations provide an estimate $\mathbf{s}^t(t)$ of $\mathbf{s}_t$ from the observations $\mathbf{x}_1, \ldots, \mathbf{x}_t$. Setting $\mathbf{s}^0(1) = \boldsymbol{\mu}_1$ and $\Sigma^0(1) = \sigma_1^2 I$, they can be written (see e.g. (Rosti & Gales, 2001)):

$$\mathbf{s}^{t-1}(t) = A\mathbf{s}^{t-1}(t-1) + \boldsymbol{\mu}_s \tag{2}$$
$$\Sigma^{t-1}(t) = A\Sigma^{t-1}(t-1)A' + \sigma_s^2 I \tag{3}$$
$$\Sigma_e(t) = C\Sigma^{t-1}(t)C' + \sigma_x^2 I \tag{4}$$
$$G_t = \Sigma^{t-1}(t)C'\Sigma_e^{-1}(t) \tag{5}$$
$$\mathbf{e}_t = \mathbf{x}_t - C\mathbf{s}^{t-1}(t) - \boldsymbol{\mu}_x \tag{6}$$
$$\mathbf{s}^t(t) = \mathbf{s}^{t-1}(t) + G_t\mathbf{e}_t \tag{7}$$
$$\Sigma^t(t) = \Sigma^{t-1}(t) - G_tC\Sigma^{t-1}(t) \tag{8}$$

The Kalman Smoother computes an estimate $\hat{\mathbf{s}}(t)$ of $\mathbf{s}_t$ from the entire observation sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$. It requires the filtering procedure to be performed and it implements the following set of equations:

$$J_{t-1} = \Sigma^{t-1}(t-1)A'(\Sigma^{t-1}(t-1))^{-1} \tag{9}$$
$$\varsigma(t-1) = \mathbf{s}^{t-1}(t-1) + J_{t-1}(\varsigma(t) - \mathbf{s}^{t-1}(t)) \tag{10}$$
$$\Gamma(t-1) = \Sigma^{t-1}(t-1) + J_{t-1}(\Gamma(t) - \Sigma^{t-1}(t))J'_{t-1} \tag{11}$$
$$\Gamma^{t-1}(t) = \Gamma(t)J'_{t-1} \tag{12}$$

starting with $\varsigma(T) = \mathbf{s}^{(T)}(T)$ and $\Gamma(T) = \Sigma^{(T)}(T)$.

2.2. Kalman Filtering with Kernels

2.2.1. Proposed Model

In order to tackle nonlinear time series processing, we propose a kernelized version of the Kalman filter. To do so, instead of directly working with the nonlinear $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, we focus on the associated series $\mathbf{x}^\phi_{1:T} = \{\mathbf{x}^\phi_1, \ldots, \mathbf{x}^\phi_T\}$, where $\mathbf{x}^\phi_t$ stands for $\phi(\mathbf{x}_t)$. The nonlinear mapping $\phi : \mathbb{R}^d \to \mathcal{H}$ is assumed to map vectors from the *input space* $\mathbb{R}^d$ to a so-called *feature space* $\mathcal{H}$, where the inner product $\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle$ can be evaluated thanks to a Mercel kernel function k such that: $\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle = k(\mathbf{x}, \mathbf{y})$ (Boser et al., 1992). The generative model we propose to study is the following:

$$\mathbf{s}^\phi_{t+1} = A^\phi\mathbf{s}^\phi_t + \boldsymbol{\mu}^\phi_s + \boldsymbol{\nu}^\phi_s \tag{13a}$$
$$\mathbf{x}^\phi_t = \mathbf{s}^\phi_t + \boldsymbol{\mu}^\phi_x + \boldsymbol{\nu}^\phi_x \tag{13b}$$

where all involved vectors are in $\mathcal{H}$ and where A^ϕ is a matrix of appropriate size to be multiplied by a vector of $\mathcal{H}$; $\boldsymbol{\nu}^\phi_s$ and $\boldsymbol{\nu}^\phi_x$ are zero-mean gaussian noise vectors of covariances $\sigma_s^2 I$ and $\sigma_x^2 I$, respectively. Actually, these noise vectors are assumed to live in the space $\mathcal{H}_x$ spanned by the vectors $\mathbf{x}^\phi_1, \ldots, \mathbf{x}^\phi_T$ and I is the identity matrix in this space. Although this might seem to be very strong an assumption, the efficiency of Kernel PCA (Mika et al., 1999) for denoising shows that such a noise model can cover a wide range of noise distributions in the input space.

As for all kernel methods, the key idea behind the model (13) is that a linear model defined in the feature space corresponds to a nonlinear one in the input space. The ability to fully exploit the model (13) should thus be connected to the ability of addressing some nonlinear time series processing.

It can be noticed that the model (13) is a generalization of the model proposed by Ralaivola and d'Alché-Buc (2004), which corresponds to the specific case $\boldsymbol{\mu}^\phi_x = 0$ and $\sigma_x^2 = 0$.

2.2.2. Kernel Filtering and Smoothing

Assume (see next subsection) that the parameters A^ϕ, $\boldsymbol{\mu}^\phi_s$, $\boldsymbol{\mu}^\phi_1$ and $\boldsymbol{\mu}^\phi_x$ are of the form $B\tilde{A}B'$, $B\tilde{\boldsymbol{\mu}}_s$, $B\tilde{\boldsymbol{\mu}}_1$ and $B\tilde{\boldsymbol{\mu}}_x$, respectively, where $B = [\mathbf{b}_1 \cdots \mathbf{b}_\ell]$ is an orthonormal basis of the space $\mathcal{H}_x$ (thus of dimension ℓ). Such a basis can be obtained from the principal axes of a Kernel PCA (Mika et al., 1999) of $\mathbf{x}^\phi_1, \ldots, \mathbf{x}^\phi_T$ having strictly positive eigenvalues, $\mathbf{x}^\phi_{1:T} = \{\mathbf{x}^\phi_1, \ldots, \mathbf{x}^\phi_T\}$ being the considered training series.

If a noisy time series $\mathbf{o}_{1:\Delta} = \{\mathbf{o}_1, \ldots, \mathbf{o}_\Delta\}$ assumed to be generated from the same model as $\mathbf{x}_{1:T}$ is observed, it is possible to perform the filtering and the smoothing procedures, extending (2)–(12) to obtain the set of *kernel filtering* and *kernel smoothing* equations thanks to some matrix algebra (the details are omitted for sake of clarity). Introducing $(\beta_t)_{t\geq 1}$, $(\alpha_t)_{t\geq 1}$ and $(\lambda_t)_{t\geq 1}$

$$\beta_1 = \sigma_1^2, \quad \text{and} \quad \beta_t = \sigma_x^2, \quad t = 2, \ldots, \Delta$$
$$\alpha_t = \frac{\beta_t}{\sigma_x^2 + \beta_t}, \quad \lambda_t = (1-\alpha_t)\beta_t, \quad t = 1, \ldots, \Delta$$

the kernel filtering equations are, for $t = 1, \ldots, \Delta$:

$$\tilde{\mathbf{s}}^{t-1}(t) = \tilde{A}\left[\alpha_{t-1}B'\phi(\mathbf{o}_{t-1}) + \tilde{\mathbf{s}}^{t-1}(t-1)\right] + \tilde{\boldsymbol{\mu}}_s$$
$$\tilde{\Sigma}^{t-1}(t) = \tilde{A}\left[\lambda_{t-1}I + \tilde{\Sigma}^{t-1}(t-1)\right]\tilde{A}'$$
$$\tilde{G}_t = (1-\alpha_t)\left[(\sigma_x^2 + \beta_t)I + \tilde{\Sigma}^{t-1}(t)\right]^{-1}\tilde{\Sigma}^{t-1}(t)$$
$$\tilde{\mathbf{s}}^t(t) = \tilde{G}_tB'\phi(\mathbf{o}_t) + \left[(1-\alpha_t)I - \tilde{G}_t\right](\tilde{s}^{t-1}(t) + \tilde{\boldsymbol{\mu}}_x) - \tilde{\boldsymbol{\mu}}_x$$
$$\tilde{\Sigma}^t(t) = -\beta_t\tilde{G}_t + \left[(1-\alpha_t)I - \tilde{G}_t\right]\tilde{\Sigma}^{t-1}(t)$$

with $\tilde{\mathbf{s}}^0(1) = \tilde{\boldsymbol{\mu}}_1$ and $\tilde{\Sigma}^0(1) = 0$. $B'\phi(\mathbf{o}_t)$ is an ℓ-dimensional vector whose computation only depends on kernel evaluations. From these equations and starting with $\tilde{\varsigma}(\Delta) = \tilde{\mathbf{s}}^\Delta(\Delta)$ and $\tilde{\Gamma}(\Delta) = \tilde{\Sigma}^\Delta(\Delta)$, the kernel smoothing equations follow readily for $t = \Delta, \ldots, 2$:

$$\begin{aligned}
H_{t-1} &= \lambda_{t-1} I + \tilde{\Sigma}^{t-1}(t-1)\\
\tilde{J}_{t-1} &= \frac{1}{\lambda_{t-1}} H_{t-1}\tilde{A}'\left[I - H_{t-1}^{-1}\tilde{\Sigma}^{t-1}(t-1)\right]\\
\tilde{\varsigma}(t-1) &= \tilde{\mathbf{s}}^{t-1}(t-1) + \tilde{J}_{t-1}\left[\alpha_t B'\phi(\mathbf{o}_t) + \tilde{\varsigma}(t) - \tilde{\mathbf{s}}^{t-1}(t)\right]\\
\tilde{\Gamma}(t-1) &= \tilde{\Sigma}^{t-1}(t-1) - \tilde{J}_{t-1}\left[\alpha_t\beta_t I + \tilde{\Sigma}^{t-1}(t) - \tilde{\Gamma}(t)\right]\tilde{J}'_{t-1}\\
\tilde{\Gamma}^{t-1}(t) &= \left[\lambda_t I + \tilde{\Gamma}(t)\right]\tilde{J}_{t-1}
\end{aligned}$$

These procedures make it possible to estimate the most probable process states and to make prediction or denoising tasks when nonlinear process are studied. The filtered value $\mathbf{o}_f^{\phi}(t)$ and the smoothed value $\mathbf{o}_s^{\phi}(t)$ of $\phi(\mathbf{o}_t)$ are respectively given by:

$$\begin{aligned}
\mathbf{o}_f^{\phi}(t) &= \alpha_t\phi(\mathbf{o}_t) + B\tilde{\mathbf{s}}^t(t) + B\tilde{\boldsymbol{\mu}}_x\\
\mathbf{o}_s^{\phi}(t) &= \alpha_t\phi(\mathbf{o}_t) + B\tilde{\varsigma}(t) + B\tilde{\boldsymbol{\mu}}_x.
\end{aligned}$$

2.2.3. Learning the Parameters

In this section, we show how the parameters of the Kernel Kalman Filter can be learned in combination with the filtering and smoothing pass by the use of an EM procedure in a way similar to that recalled by Rosti and Gales (2001).

Recall that the EM algorithm for linear dynamical systems based on a training sequence $\mathbf{x}_{1:T}$ aims at maximizing the likelihood $\mathcal{L}_{lds}$

$$\begin{aligned}
\mathcal{L}_{lds}(\mathbf{x}_{1:T}|\boldsymbol{\theta}) = &-\frac{d}{2}\log\sigma_1^2 - \frac{1}{2\sigma_1^2}(\mathbf{x}_1 - \boldsymbol{\mu}_1)'(\mathbf{x}_1 - \boldsymbol{\mu}_1)\\
&-\frac{d(T-1)}{2}\log\sigma_s^2 - \frac{dT}{2}\log\sigma_x^2\\
&-\frac{1}{2\sigma_s^2}\sum_{t=2}^{T}\|\mathbf{s}_t - A\mathbf{s}_{t-1} - \boldsymbol{\mu}_s\|^2\\
&-\frac{1}{2\sigma_x^2}\sum_{t=2}^{T}\|(\mathbf{x}_t - C\mathbf{s}_t - \boldsymbol{\mu}_x)\|^2.
\end{aligned}$$

The EM algorithm to maximize this likelihood requires the implementation of the two following steps:

E step: perform the filtering and smoothing procedures given the parameter $\boldsymbol{\theta}$ with the observed (training) sequence $\mathbf{x}_{1:T}$;

M step: maximize the expectation of $\mathcal{L}_{lds}$ with respect to $\boldsymbol{\theta}$ using the filtered and smoothed values and the values $R(t) = \Gamma(t) + \varsigma(t)\varsigma'(t)$ and $R^{t-1}(t) = \Gamma^{t-1}(t) + \varsigma(t)\varsigma'(t-1)$ (see (Rosti & Gales, 2001) for more details).

This EM procedure remains valid when the learning of the parameters of a Kernel Kalman Filter is addressed. Given a basis B of $\mathcal{H}_x$, the filtering and the smoothing pass described above must first be implemented.

To perform the M step, it may be useful to compute the values $\tilde{R}(t)$, $\tilde{R}^{t-1}(t)$ and the auxiliary vectors $\tilde{\mathbf{c}}_t$:

$$\begin{aligned}
\tilde{\mathbf{c}}_t &= \alpha_t B'\mathbf{x}_t^{\phi} + \tilde{\varsigma}(t)\\
\tilde{R}(t) &= \tilde{\Gamma}(t) + \tilde{\mathbf{c}}_t\tilde{\mathbf{c}}_t'\\
\tilde{R}^{t-1}(t) &= \tilde{\Gamma}^{t-1}(t) + \tilde{\mathbf{c}}_t\tilde{\mathbf{c}}_{t-1}'
\end{aligned}$$

These values and subsequent values rely on the assumption that each vector $\mathbf{s}_t^{\phi}$ of (1a) is searched for as an element of $\mathcal{H}_x$. In addition, let the vectors $\mathbf{f} = \sum_{t=2}^{T}\tilde{\mathbf{c}}_t$ and $\mathbf{g} = \sum_{t=2}^{T}\tilde{\mathbf{c}}_{t-1}$ together with the matrices

$$\begin{aligned}
G &= \sum_{t=2}^{T}\tilde{R}(t-1), \qquad R = G - \mathbf{g}\mathbf{g}'/(T-1)\\
L &= \sum_{t=2}^{T}\tilde{R}^{t-1}(t), \qquad S = L - \mathbf{f}\,\mathbf{g}'/(T-1).
\end{aligned}$$

According to the values of A and $\boldsymbol{\mu}_s$ obtained when a linear dynamical system is considered, we get:

$$\tilde{A} = \frac{1}{\sum_{t=2}^{T}\lambda_{t-1}} S\left[I - \left((\sum_{t=2}^{T}\lambda_{t-1})I + R\right)^{-1} R\right] \tag{14}$$

$$\tilde{\boldsymbol{\mu}}_s = \frac{1}{T-1}(\mathbf{f} - \tilde{A}\mathbf{g}) \tag{15}$$

$$\tilde{\boldsymbol{\mu}}_1 = \tilde{\mathbf{c}}_1 \tag{16}$$

from which $A = B\tilde{A}B'$, $\boldsymbol{\mu}_s^{\phi} = B\tilde{\boldsymbol{\mu}}_s$ and $\boldsymbol{\mu}_1^{\phi} = B\tilde{\boldsymbol{\mu}}_1$. In order to cope with the possible high complexity induced by the use of kernels, it may be sensible to put a gaussian prior p_γ

$$p_\gamma(A) \propto \exp(-\frac{\gamma}{2\sigma_s^2}\,\mathrm{tr}\,(A'A)) \tag{17}$$

on A, favoring A with small entries. Resorting to such control complexity, each occurence of $\sum_{t=2}^{T}\lambda_{t-1}$ in (14) must be replaced by $\sum_{t=2}^{T}\lambda_{t-1} + \gamma$.

For σ_s^2 and σ_1^2 we have:

$$\begin{aligned}
\sigma_s^2 &= \frac{1}{T-1}\sum_{t=2}^{T}\lambda_{t-1} + \frac{1}{\ell(T-1)}\,\mathrm{tr}\,\left(F - \tilde{A}L - \tilde{\boldsymbol{\mu}}_s\mathbf{f}'\right)\\
\sigma_1^2 &= \lambda_1 - \frac{1}{\ell}\tilde{\mathbf{c}}_1'\tilde{\mathbf{c}}_1.
\end{aligned}$$

Finally, the values for $\boldsymbol{\mu}_x^{\phi}$ and σ_x^2 are given by:

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_x &= \frac{1}{T}\sum_{t=1}^{T}((1-\alpha_t)B'\mathbf{x}^{\phi} - \tilde{\varsigma}(t))\\
\sigma_x^2 &= \frac{1}{\ell T}\sum_{t=1}^{T}\left((1-\alpha_t)\mathbf{x}_t^{\phi\prime}BB'\mathbf{x}_t^{\phi} - \mathbf{x}_t^{\phi\prime}B(\tilde{\varsigma}(t) + \tilde{\boldsymbol{\mu}}_x)\right).
\end{aligned}$$

from which $\boldsymbol{\mu}_x^\phi = B\tilde{\boldsymbol{\mu}}_x$ (note that $\mathbf{x}_t^{\phi\prime} BB'\mathbf{x}_t^\phi = k(\mathbf{x}_t, \mathbf{x}_t)$). Hence, the assumption that each $\mathbf{s}_t^\phi$ is in $\mathcal{H}_x$ validates the form used by the kernel filtering and smoothing procedures (cf. (14)–(16)).

2.3. Finding the Preimages through Learning

The model (1) only deals with vectors in $\mathcal{H}$ whereas vectors from the input space $\mathbb{R}^d$ are often needed. Given a vector $\mathbf{z}^\phi$ in $\mathcal{H}$, finding a good vector $\mathbf{x}$ in $\mathbb{R}^d$ such that $\phi(\mathbf{x})$ is as close as possible to $\mathbf{z}^\phi$ is known as the *preimage* problem.

In order to solve this problem, we employ an original learning strategy presented in (Ralaivola & d'Alché-Buc, 2004): instead of trying to use a strategy requiring to solve some mathematical problem each time a preimage is needed, we consider the training set $\{(B'\mathbf{x}_1^\phi, \mathbf{x}_1), \ldots, (B'\mathbf{x}_T^\phi, \mathbf{x}_T)\}$ from which a regressor is learned. In other words, for each $\mathbf{x}_t$ we learn the mapping between the coordinate vector of $\phi(\mathbf{x}_t) \in \mathcal{H}_x$ with respect to B to itself. The learned regressor provides an efficient tool to estimate the preimages.

3. Related work

This work is related to the extended versions of the classical Kalman filter, namely the Extended Kalman Filter or EKF (Anderson & Moore, 1979; Welch & Bishop, 1995) and the Unscented Kalman Filter or UKF (Julier & Uhlmann, 1997; van der Merwe et al., 2001) and their variants. These algorithms can indeed be used in tasks involving prediction and/or noise removal when nonlinear processes are studied. However, as evidenced previously, our model does not need to resort to any approximated estimation strategy while, in addition, it makes it possible to implement the learning of the model parameters.

Two other works are closely related to ours. The first one is the work of Suykens et al. (2001) who tackled the so-called N-stage optimal control problem, a subproblem of the nonlinear filtering problem, using support vector regressors to model the transition process. The second one is that of Engel et al. (2002), who proposed to learn the weight vector of a kernel regressor using a Kalman filter with a fixed state transition process.

The topic of preimages determination is an issue of importance since the appearance of the Kernel PCA algorithm. An algorithm has been recently proposed by Kwok and Tsang (2003) to determine the preimages. This algorithm is based on the fact that some kernels can be 'inverted' to provide the distance in input space between a pair of vectors from their kernel value. This algorithm finds the location of the preimages based on distance constraints in the feature space. Bakir et al. (2004) have proposed simultaneously to use a way to compute the preimages resorting to a learning strategy and have shown the efficiency of this approach.

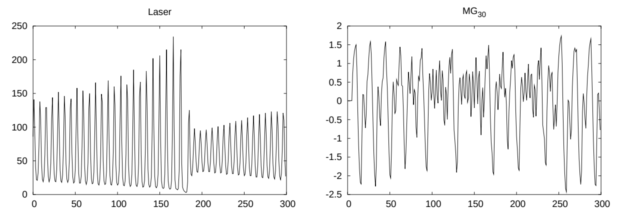

Figure 1. Left, the fi rst 300 points of the *Laser* time series. Right, 300 points of the series MG_{30}.

Eventually, regarding the issue of parameters learning, our algorithm must be related to the work of Ghahramani and Roweis (1999) and Briegel and Tresp (1999). The algorithms presented in these papers however assume some semi-parametric models whose parameters have to be learned. The approach that we have adopted is different in the sense that the proposed EM automatically determines the appropriate transition model from (14).

4. Experiments

4.1. Datasets and Protocol

The experiments presented in this section have been carried out on four datasets: the univariate time series *Laser* (see Figure 1, left) and *Mackey-Glass* MG_{30} (see Figure 1, right) and the multi-dimensional time series *Ikeda* and *Lorenz*. Two kinds of KKF performances are evaluated: the prediction capacity and the ability of processing noisy nonlinear time series.

The *Ikeda map* (see Figure 2, left) is related to a laser dynamics. This time series is defined from a starting point $(x_1(0), x_2(0))$ by

$$\begin{cases} \omega(t) = c_1 - \dfrac{c_3}{1 + x_1^2(t) + x_2^2(t)} \\ x_1(t+1) = r + c_2\left(x_1(t)\cos\omega(t) - x_2(t)\sin\omega(t)\right) \\ x_2(t+1) = c_2\left(x_1(t)\sin\omega(t) + x_2(t)\cos\omega(t)\right) \end{cases}$$

where c_1, c_2, c_3 and r are real valued parameters; we considered $c_1 = 0.4$, $c_2 = 0.84$, $c_3 = 6.0$, $r = 1.0$ and $\mathbf{x}(0) = [x_1(0)\ x_2(0)]' = [1.0\ 0.001]'$.

A *Lorenz* attractor (see Figure 2, middle and right) is the solution of the system of differential equations

$$\begin{cases} \dfrac{dx(t)}{dt} = -ax + ay \\ \dfrac{dy(t)}{dt} = -xz + rx - y \\ \dfrac{dz(t)}{dt} = xy - bz \end{cases}$$

We set $a = 10, r = 28$ and $b = 8/3$.

The *Laser* and *Mackey-Glass* MG_{30} time series are univariate and, as proposed by Mukherjee et al. (1997) and Müller et al. (1997), we introduce the embedding vectors

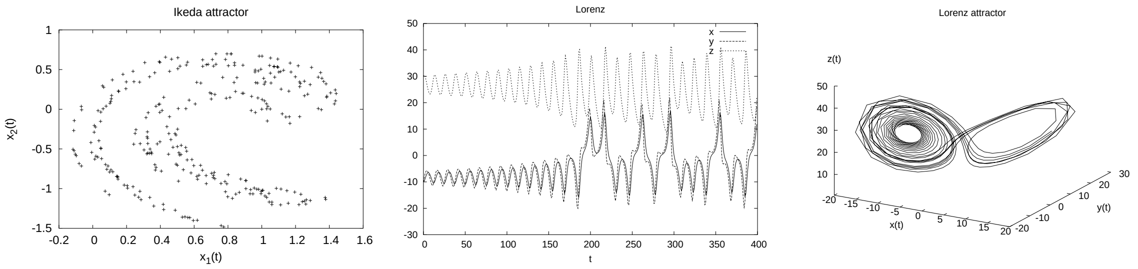

Figure 2. Left, the first 300 points of the series *Ikeda* (x_2 is plotted as a function of x_1). Middle, the first 400 points of the *Lorenz* attractor: each of the coordinate x, y and z over time and, right, the corresponding trajectory.

$\mathbf{x}_t = [x_{t-(d-1)\kappa} \cdots x_{t-\kappa}\ x_t]'$ where κ is a fixed step size; the series $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ is then used as a training series. For the *Laser* time series, we set $\kappa = 1$ and $d = 8$ whereas $\kappa = 6$ and $d = 6$ are used for *MG*$_{30}$ (those are common values for these parameters). Conversely, *Ikeda* and *Lorenz* are multi-dimensional first-order serie and the resort to embedding vectors (though possible) is not considered.

For all the datasets, the data are split up in $n_{train} = 300$ training data, $n_{test} = 200$ test data to choose the hyperparameters of the models and $n_{valid} = 300$ validation data. Three training methods for the prediction of time series are tested: a multilayer perceptron (MLP), a support vector regressor (SVR) and a Kernel Kalman Filter (KKF). For those last two kernel machines, gaussian kernels $k(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma_k^2})$ have been used. The nonlinearity used in the MLPs is implemented by a $\tanh$ function and the code used to implement the MLP and other filtering techniques is that of R. van der Werve's ReBEL toolbox[1]. The training of multi-dimensional time series with support vector regressors is done by decomposing the learning problem in as many number of training problem as the dimension of the time series.

For all the experiments, the regression machine used to determine the preimages is a Kernel Ridge Regressor with a gaussian kernel whose width is determined with respect to the performance measured on the test set.

The measurement error used to evaluate the performance of the different algorithms is the mean squared error err defined for a machine f and a series $\mathbf{x}_{1:T}$ as:

$$err(f, \mathbf{x}_{1:T}) = \frac{1}{(T-1)} \sum_{t=2}^{T} \|\mathbf{x}_t - f(\mathbf{x}_{t-1})\|^2.$$

4.2. Prediction Accuracy of KKF

Two kinds of prediction capacities are evaluated for KKF. The first one is the *one-step* ahead prediction ability, which merely consists in realizing the prediction at time $t+1$ from the actual observation at time t. The second one is the *multi-step* ahead or *trajectory* prediction prediction where the model relies on its own output estimates to compute future observations.

The training sequences provided to the algorithms are clean (non noisy) sequences. Thus, for the prediction task, KKF is learned with $\boldsymbol{\mu}_x^\phi = 0$ and $\sigma_x^2 = 0$.

Table 1 reports the validation mean squared error obtained by the different models tested for the one-step prediction and the trajectory prediction. For KKF, the complexity control discussed above (see (17)) has been tested: the results are shown for different values of γ in order to assess its influence on the prediction quality.

The analysis of the results leads to the first conclusion that KKF learning with no measurement noise is able to provide a predictor of quality comparable with that of MLP and SVR. This prediction capacity is witnessed for all the datasets even when the difficult task of the trajectory prediction is considered. This observation is all the more striking that the determination of the preimages is an approximate process and the accumulation of such approximation errors over iterated predictions may have been expected to deteriorate the overall performance.

A second observation relates to the dependence of our algorithm on the regularization γ: the unregularized method ($\gamma = 0$) never obtains the best error, neither for the one-step ahead prediction nor for the trajectory prediction. This evidences the excessive complexity induced by the use of kernel functions and, as a possible consequence, the presence of an over-fitting phenomenon. Conversely, when the regularization is too important (too large coefficient γ), the model cannot reach interesting performances.

Incidentally, a strange result about the prediction of the series *Ikeda* by our algorithm may be noticed: the best regularization parameters (with respect to the validation error) for the trajectory prediction is the one which has the worst results for the one-step prediction. Actually, if the

[1]http://choosh.ece.ogi.edu/rebel/index.html.

Table 1. Mean squared error for the one-step ahead prediction. γ is the regularizing parameter of Equation (17).

		One-step prediction				**Multi-step prediction**			
Algo.	**Param.**	***Laser***	***MG*$_{30}$**	***Ikeda*** **($\times 10^{-4}$)**	***Lorenz***	***Laser***	***MG*$_{30}$**	***Ikeda***	***Lorenz***
MLP	–	1.4326	0.0461	**7.093**	0.2837	4450	**1.046**	0.9153	486.4
SVR	–	0.2595	0.0313	8.103	**0.1811**	**2263**	1.173	0.9231	**411.8**
KKF	$\gamma = 0.00$	0.2812	0.0512	7.922	0.3134	2304	1.242	0.9705	422.2
	$\gamma = 10^{-9}$	0.2641	0.0453	7.914	0.3133	2297	1.121	0.9415	425.5
	$\gamma = 10^{-7}$	**0.2325**	0.0364	7.773	0.3133	2270	1.095	0.9914	428.1
	$\gamma = 10^{-5}$	0.2325	**0.0307**	8.421	0.3186	**2263**	1.090	1.0593	430.9
	$\gamma = 10^{-3}$	0.2732	0.0315	15.71	0.3230	2281	1.085	0.9528	441.7
	$\gamma = 10^{-1}$	0.2812	0.0374	30.81	0.7922	2298	1.111	0.8550	459.4
	$\gamma = 1.00$	0.3101	0.0521	35.65	5.1135	2315	1.241	**0.7263**	508.1

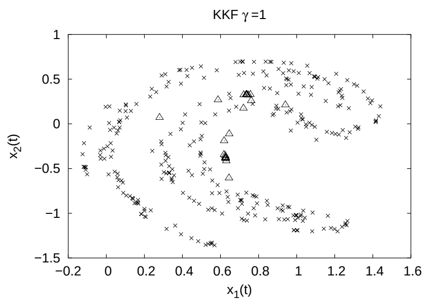

Figure 3. Trajectory prediction made by KKF ($\gamma = 1$). The triangles correspond to the KKF prediction and the crosses to the actual time series (see text for comments).

forecasted trajectory provided by the corresponding model ($\gamma = 1$) is compared to the true *Ikeda* trajectory, it appears that this low quadratic error does not square with an actual prediction ability; it is instead a sort of degenerated quasi-periodical trajectory passing through particular position (see Figure 3). This phenomenon is also observed for the values $\gamma = 0.1$ and $\gamma = 0.001$ whereas for the other values of γ the trajectory envisaged is very similar with the real trajectory (see Figure 4). The trajectory predictions for the other time series do not undergo such an atypical behaviour.

These prediction experiments validate both the efficiency of the Kernel Kalman Filter and the learning strategy to compute the preimages.

4.3. Noise Extraction

The task tackled here consists in removing the noise of one series to which an observation noise has been added. For our experiments, we added a zero-mean gaussian noise of variance σ^2_{noise} to the validation series of length n_{valid};

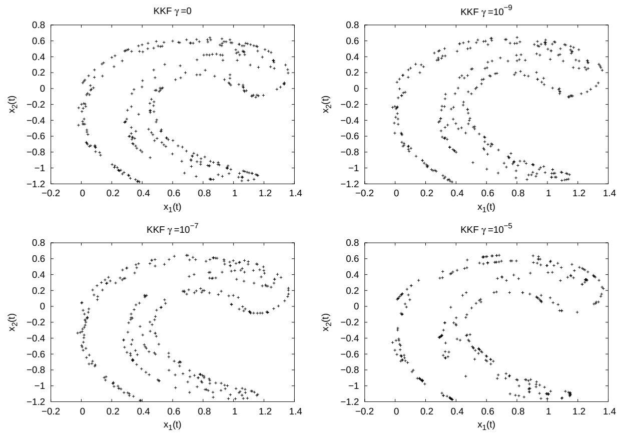

Figure 4. Trajectory prediction made by KKF on the Ikeda series (above each fi gure, the value of γ).

such a noise variance leads to a signal to noise ratio of 3dB.

The experiments described in this section have been carried out on the series *MG*$_{30}$, *Ikeda* and *Lorenz*. The parameters of KKF model are, for each time series, those which correspond to the best one-step ahead prediction results . The other filtering algorithms make use of the best MLP to model the transition between the states $\mathbf{s}_t$ (see Table 1). Each experiment of noise removal is repeated 20 times and Table 2 reports the mean-squared errors and standard deviations between the denoised signal and the original one.

The results of this table show that our method of filtering is as effective as the usually implemented methods when the problem of nonlinear series denoising is tackled. In particular, it may be noticed that in the case of *MG*$_{30}$, the measured mean squared error is lower than that provided by the others algorithms. Besides, in all cases, KKF produces results on average better than those of the usually

Table 2. Mean squared errors and standard deviation for the denoising task. CDKF differs from UKF by the strategy used to determine the so-called *Sigma* points.

Algo.	***MG***$_{30}$	***Ikeda*** **($\times 0.1$)**	***Lorenz*** **($\times 10$)**
EKF	$631.7(1230)$	$6.842(1.520)$	$111.1(213.1)$
UKF	$9.288(0.560)$	$1.624(0.086)$	$2.876(5.954)$
CDKF	$10.12(1.510)$	$1.387(0.095)$	$1.945(1.894)$
KKF	$8.741(0.435)$	$2.145(0.012)$	$20.22(31.41)$

used procedure of the Extended Kalman Filter. Regarding the *Lorenz* series, a relatively important difference between the results of our method and the best results can be noticed; the reported standard deviation however prevents from concluding about any kind of limitation regarding KKF for the *Lorenz* problem.

Finally, the main idea of these experiments of noise extraction is that our model makes it possible to proceed effectively to this task by using the matrix equations of recursive estimation given in section 2.2.2.

4.4. Dynamics Extraction

The last capacity of KKF we evaluate relates to the effectiveness of the training procedure that we described in section 2.2.3. We focus on the *Ikeda* problem and proceed as follows. First, we produce a series $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ of 300 training noisy points from the *Ikeda* map, using the transition process

$$\begin{cases} s_1(0), s_2(0) \\ \omega(t) = c_1 - \dfrac{c_3}{1 + s_1^2(t) + s_2^2(t)} \\ s_1(t+1) = r + c_2\left(s_1(t)\cos\omega(t) - s_2(t)\sin\omega(t)\right) + \nu_1^s \\ s_2(t+1) = c_2\left(s_1(t)\sin\omega(t) + s_2(t)\cos\omega(t)\right) + \nu_2^s \end{cases}$$

and the observation process

$$\left\{ \mathbf{x}_t = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} s_1(t) + \nu_1^x \\ s_2(t) + \nu_2^x \end{bmatrix} \right.$$

where the vectors $\boldsymbol{\nu}_s = [\nu_1^s \ \nu_2^s]'$ and $\boldsymbol{\nu}_x = [\nu_1^x \ \nu_2^x]'$ are noise vectors. Then, we provide KKF with the series $\mathbf{x}_{1:T}$ from which the learning procedure determines the optimal parameters. Lastly, we measure the one-step ahead prediction capacity on a clean validation series of length 300.

In order to show the genericity of our model, the kernel used here is no longer gaussian but polynomial of degree 5 such that $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^5$. A value of regularization $\gamma = 10^{-5}$ is used.

For the experiments, $\boldsymbol{\nu}_x$ is selected as a zero-mean gaussian vector with variance $\sigma_x^2 = 0.13$, which corresponds to the use of a signal to noise ratio of 3dB with respect to the validation time series. Three types of noise are used

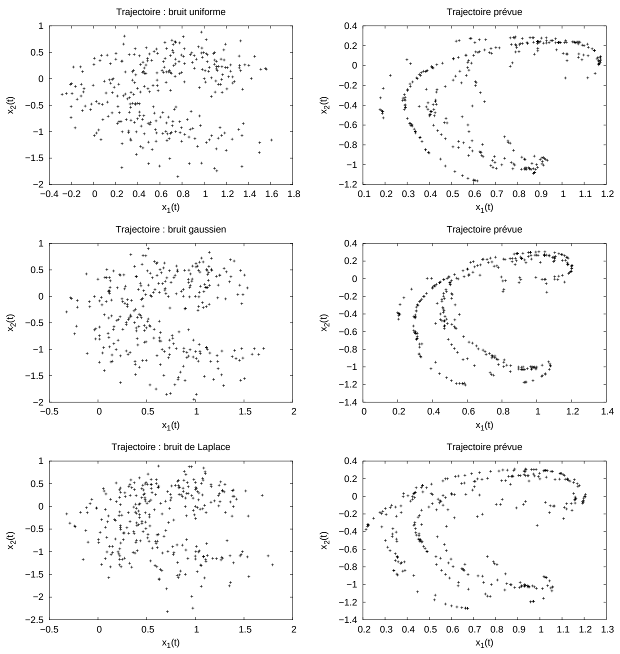

Figure 5. Left column: noisy training series. Right column: extracted dynamics. The first row corresponds to a uniform noise (validation error: 0.061), the second one to a gaussian noise (error: 0.061) and the last one to a Laplace noise (error: 0.064).

for $\boldsymbol{\nu}_s$: a zero-mean gaussian noise with variance $\sigma_x^2 = 0.01$ (Figure 5, top), a uniform noise on $[-0.2; 0.2]$ (Figure 5, middle) and a zero-mean Laplace noise distributed as $p(x) = \exp(-|x|/b)$ with $b = 0.044721$ (Figure 5, bottom); choosing b this way leads to a noise of variance 0.01.

Figure 5 illustrates the efficiciency of our method for the extraction of complex dynamics from a noisy series. In particular, it must be noticed that our algorithm is able to extract dynamics very similar to the actual dynamics whichever noise model is considered. These results are all the more remarkable as the studied dynamics are very complex, which again validates our kernel extension of linear dynamical systems.

5. Conclusion and Future Work

We have presented a new kernel method, the *Kernel Kalman Filter*, which is an extension of the Kalman filter and linear dynamical systems. We have set up the filtering, smoothing and learning procedures for this model showing that the involved equations are closely related to their classical linear counterparts. Throughout various experiments, we have shown the efficiency of KKF in tasks such as nonlinear time series denoising and prediction; the problem of extracting complex dynamics from noisy nonlinear dynamics has also successfully been tackled by our approach. Lastly, the presented results confirm the ability of the preimage learning strategy to determine good preimages.

Several extensions to this work might be envisioned. For instance, the order of the matrices involved by the model linearly scales with the length of the studied time series, leading to possible heavy computations and/or storage memory problems. A sparse incremental greedy method could be a workaround to this problem. Besides, whereas working with a linear process in the state space allows some interpretability, this advantage is lost when working in the feature space. Thus, an interesting extension of this work would be to use kernels well-suited for interpretability and to learn their parameters in order to extract valuable knowledge. Finally, the proposed algorithm offers an original framework to model time series of non vectorial objects, provided that a suitable kernel is available.

References

Anderson, B. D., & Moore, J. B. (1979). *Optimal filtering*. Englewood Cliffs, NJ: Prentice Hall.

Bakir, G. H., Weston, J., & Schölkopf, B. (2004). Learning to Find Pre-Images. *Adv. in Neural Information Processing Systems*.

Boser, B., Guyon, I., & Vapnik, V. (1992). A Training Algorithm for Optimal Margin Classifiers. *Proc. of the 5th Workshop on Comp. Learning Theory*.

Briegel, T., & Tresp, V. (1999). Fisher Scoring and a Mixture of Modes Approach for Approximate Inference and Learning in Nonlinear State Space Models. *Adv. in Neural Information Processing Systems*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistics Society*, *39*.

Engel, Y., Mannor, S., & Meir, R. (2002). Sparse online greedy support vector regression. *Proc. of the 13th European Conference on Machine Learning*.

Ghahramani, Z., & Roweis, S. (1999). Learning nonlinear dynamical systems using an em algorithm. *Adv. in Neural Information Processing Systems*.

Julier, S., & Uhlmann, J. (1997). A New Extension of the Kalman Filter to Nonlinear Systems. *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, *82*, 35–45.

Kwok, J. T., & Tsang, I. W. (2003). The pre-image problem in kernel methods. *Proc. of the 20th International Conference on Machine Learning*.

Mika, S., Schölkopf, B., Smola, A. J., Müller, K.-R., Scholz, M., & Rätsch, G. (1999). Kernel PCA and De-Noising in Feature Spaces. *Adv. in Neural Information Processing Systems*.

Mukherjee, S., Osuna, E., & Girosi, F. (1997). Nonlinear prediction of chaotic time series using support vector machines. *Proc. of IEEE NNSP'97*.

Müller, K., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V. (1997). Predicting Time Series with Support Vector Machines. *Proc. of Int. Conf. on Artificial Neural Networks*.

Ralaivola, L., & d'Alché-Buc, F. (2004). Dynamical Modeling with Kernels for Nonlinear Time Series Prediction. *Adv. in Neural Information Processing Systems*.

Rosti, A.-V., & Gales, M. (2001). *Generalised linear Gaussian models* (Technical Report CUED/F-INFENG/TR.420). Cambridge University Engineering Department.

Suykens, J. A. K., Vandewalle, J., & Moor, B. D. (2001). Optimal control by least squares support vector machines. *Neural Networks*, *14*, 23–35.

van der Merwe, R., de Freitas, N., Doucet, A., & Wan, E. (2001). The Unscented Particle Filter. *Adv. in Neural Information Processing Systems*.

Welch, G., & Bishop, G. (1995). *An introduction to the Kalman filter* (Technical Report TR 95-041). University of North Carolina.