Dynamical Modeling with Kernels for Nonlinear Time Series Prediction

Liva Ralaivola  
Laboratoire d'Informatique de Paris 6  
Université Pierre et Marie Curie  
8, rue du capitaine Scott  
F-75015 Paris, FRANCE  
liva.ralaivola@lip6.fr

Florence d’Alché–Buc  
Laboratoire d'Informatique de Paris 6  
Université Pierre et Marie Curie  
8, rue du capitaine Scott  
F-75015 Paris, FRANCE  
florence.dalche@lip6.fr

Abstract

We consider the question of predicting nonlinear time series. Kernel Dynamical Modeling, a new method based on kernels, is proposed as an extension to linear dynamical models. The kernel trick is used twice: first, to learn the parameter of the model, and second, to compute preimages of the time series predicted in the feature space by means of Support Vector Regression. Our model shows strong connection with the classic Kalman Filter model, with the kernel feature space as hidden state space. Kernel Dynamical Modeling is tested against two benchmark time series and achieves high quality predictions.

1 Introduction

Prediction, smoothing and filtering are traditional tasks applied to time series. Machine learning community has recently paid a lot of attention to these problems and especially to nonlinear time series prediction in various areas such as biological signals, speech [2] or financial markets. To cope with non linearities, extensions of Kalman Filtering [5, 15], have been proposed for filtering and smoothing while recurrent artificial neural networks [4] and support vector regressors [9, 10] have been developed for prediction purposes. In this paper, we focus on prediction tasks and introduce a powerful method based on the kernel trick [3], which has been successfully used in tasks ranging from classification [3, 7] and regression [13, 14] to data analysis [1, 8]. Time series modeling is addressed by extending the framework of observable linear dynamical systems [12] to the feature space defined by a kernel. Prediction is realized in the feature space and then is transformed to obtain corresponding preimages in the input space. While the proposed model could be used for smoothing as well as filtering, we focus here on the prediction goal. Links to Kalman filter model [6] can be drawn by noticing that the time serie in the feature space plays the role of the hidden process underlying the Kalman filter. The paper is organized as follows. In the first section, after a brief introduction to the problem we describe how modeling can take place in the feature space. Then we explain how to go back to the input space by solving the preimage problem. In the second section we present results in prediction for our model and a Support Vector Regressor used as an auto-regressive model. In the fourth section, the estimation algorithm is discussed and the link of the model to Kalman filter is highlighted. We finally conclude by giving some further perspectives to our work.
2 Principles of Dynamical Modeling with Kernels

2.1 Basic Formulation

We want to address the problem of predicting \( d \)-dimensional nonlinear real-valued time series defined as
\[
x_{t+1} = h(x_t) + u
\]
where \( h \) is a nonlinear function and \( u \) a noise vector. A usual way to model such a series can be done with the help of recurrent neural networks [4] or a support vector machine [9].

In this work, we propose to deal with this problem by extending linear dynamical modeling thanks to the kernel trick. Instead of considering the observation sequence \( x_{1:T} = \{x_1, \ldots, x_T\} \), we consider the sequence \( x_{1:T}^\phi = \{\phi(x_1), \ldots, \phi(x_T)\} \), where \( \phi \) is a mapping from \( \mathbb{R}^d \) to \( \mathcal{F} \) and \( k \) its associated kernel function [14] such that \( k(v_1, v_2) = \langle \phi(v_1), \phi(v_2) \rangle \) \( \forall v_1, v_2 \in \mathbb{R}^d \), \( \langle \cdot, \cdot \rangle \) being the inner product of \( \mathcal{F} \). The model to study can then be written as:
\[
x_{t+1}^\phi = A^\phi x_t^\phi + \mu^\phi + \nu^\phi
\]
where \( A^\phi \) is the process transition matrix, \( \mu^\phi \) an offset vector, \( \nu^\phi \in \mathcal{F} \) a gaussian isotropic noise of magnitude \( \sigma^2 \) and where \( x_t^\phi \) stands for \( \phi(x_t) \).

We are going to show that it is possible to apply the likelihood maximum principle to identify \( \sigma^2 \), \( A^\phi \) and \( \mu^\phi \) and come back to the input space thanks to preimages determination.

2.2 Estimation of the Model Parameters

Learning the parameters of the model (2) by maximum likelihood given an observation sequence \( x_{1:T}^\phi \) merely consists in optimizing the associated log-likelihood \( \mathcal{L}^\phi(x_{1:T}^\phi; \theta^\phi) \):
\[
\mathcal{L}^\phi(x_{1:T}^\phi; \theta^\phi) = \ln \left( \prod_{t=2}^{T} P(x_t^\phi \mid x_{1:t-1}^\phi) \right)
\]
\[
= h(\mu_1^\phi, \Sigma_1^\phi) - \frac{1}{2\sigma^2} \sum_{t=2}^{T} \|x_t^\phi - A^\phi x_{t-1}^\phi - \mu^\phi\|^2 - \frac{1}{2}p(T-1) \ln \sigma^2
\]
where \( p \) is the dimension of \( \mathcal{F} \), \( h(\mu_1^\phi, \Sigma_1^\phi) \) is a function straightforward to compute which we let aside as it does not add any complexity in setting the gradient of \( \mathcal{L}^\phi \) to \( \mathbf{0} \). Indeed, performing this task leads to the equations:
\[
A^\phi = \left( \sum_{t=2}^{T} x_t^\phi x_{t-1}^\phi - \frac{1}{T-1} \sum_{t=2}^{T} x_t^\phi x_{t-1}^\phi \right) \left( \sum_{t=2}^{T} x_{t-1}^\phi x_{t-1}^\phi - \frac{1}{T-1} \sum_{t=2}^{T} x_{t-1}^\phi x_{t-1}^\phi \right)^{-1}
\]
\[
\mu^\phi = \frac{1}{T-1} \sum_{t=2}^{T} (x_t^\phi - A^\phi x_{t-1}^\phi)
\]
\[
\sigma^2 = \frac{1}{p(T-1)} \sum_{t=2}^{T} \|x_t^\phi - A^\phi x_{t-1}^\phi - \mu^\phi\|^2
\]
\[1\]We use the simplifying notation \( \theta^\phi := \{A^\phi, \mu^\phi, \sigma^2, \mu_1^\phi, \Sigma_1^\phi\} \), where \( \mu_1^\phi \) and \( \Sigma_1^\phi \) are the parameters of the first gaussian observation \( x_1^\phi \).
which need to address two problems: inverting a matrix which could be of infinite dimension (e.g., if a gaussian kernel is used) and/or singular (equation (3)) and making a division by the dimension of the feature space ($p$ in equation (5)).

A general solution to circumvent these problems is to introduce an orthonormal basis $U = \{u_1^\phi, \ldots, u_m^\phi\}$ for the subspace $\mathcal{F}_x$ of $\mathcal{F}$ spanned by $x_1^\phi:T$. For instance, $U$ can be obtained by computing the set of principal components with non-zero eigenvalues of $x_1^\phi:T$ following the procedure proposed in [8]. Once such a set of vectors is available, trying to find good parameters for the model (2) is equivalent to finding an $m$-dimensional linear dynamical model for the sequence $z_{1:T} = \{z_1, \ldots, z_T\}$ where $z_t$ is the vector of coordinates of $x_t^\phi$ with respect to $U$, i.e.:

$$z_t = \left[\langle x_t^\phi, u_1^\phi \rangle \langle x_t^\phi, u_2^\phi \rangle \ldots \langle x_t^\phi, u_m^\phi \rangle \right]' \quad \forall t = 1, \ldots, T. \quad (6)$$

Given $z_{1:T}$, the following linear dynamical model has to be considered:

$$z_{t+1} = A_z z_t + \mu_z + \nu_z \quad (7)$$

where $\nu_z$ is again a gaussian noise vector of variance $\sigma^2$. Determining a basis of $\mathcal{F}_x$ allows to learn the linear dynamical model (7). As it is based on the coordinates of the observed vectors $x_1^\phi, \ldots, x_T^\phi$ with respect to the basis, it is equivalent to learning (2). The parameters are estimated thanks to equations (3), (4) and (5) where $x_t^\phi$ is replaced with $z_t$ and $p$ with $m$.

For the sake of generalization ability, it might be useful to choose $A_z$ as simple as possible [14]. To do this, we put a penalization on matrices $A_z$ having large values, by imposing a prior distribution $p_A$ on $A_z$ defined as: $p_A(A_z) \propto \exp(-\frac{\gamma}{2} \text{trace}(A_z'A_z))$, $\gamma > 0$. The computation of the maximum a posteriori values for $A$, $\mu$ and $\sigma^2$ is very similar to (3), (4) and (5) except that a few iterations of gradient ascent have to be done.

### 2.3 Back to the Input Space: the Preimage Problem

Predicting the future observations with model (7) gives vectors in the feature space $\mathcal{F}$ while vectors from the input space $\mathcal{R}^d$ are needed. Given a vector $z^\phi$ in $\mathcal{F}$, finding a good vector $x$ in $\mathcal{R}^d$ such that $\phi(x)$ is as close as possible to $z^\phi$ is known as the preimage problem.

Mika et al. [8] propose to tackle this problem with considering the optimization problem:

$$\min_x \|\phi(x) - z^\phi\|^2.$$

This problem can be solved efficiently by gradient descent techniques for gaussian kernels. Nevertheless, it may need several optimization phases with different starting points to be ran in the case of other kernels (e.g. polynomial kernels of some particular degree).

Here, we propose to use Support Vector Regression (SVR) to solve the preimage problem. This avoids any local minimum problem and allows to benefit from the fact that we have to work with vectors from the inner product space $\mathcal{F}$.

#### 2.3.1 SVR and Preimages Learning

Given a sample dataset $\mathcal{S} = \{(z_1, y_1), \ldots, (z_n, y_n)\}$ with pairs in $\mathcal{Z} \times \mathcal{R}$, the SVR algorithm assumes a structure on $\mathcal{Z}$ given by a kernel $k_z$ and its associated mapping $\phi$ and feature space $\mathcal{F}$ (see [14]). It proceeds as follows (see [13] and [14] for further details). Given a real positive value $\epsilon$, the algorithm determines a function $f$ such that (a) it maps each $z_i$ to a value not having deviation larger than $\epsilon$ from $y_i$, and (b) it is as flat as possible.
This function computes its output as \( f(z) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) k_z(z_i, z) + b \) where the vectors \( \alpha^* \) and \( \alpha \) are the solutions of the quadratic problem
\[
\begin{align*}
\max_{\alpha^*, \alpha} & \quad -\varepsilon'(\alpha^* + \alpha) + y'(\alpha^* - \alpha) - \frac{1}{2} \left( (\alpha^* - \alpha) K(Z) (\alpha^* - \alpha) + \frac{1}{C}(\alpha^* \alpha^* + \alpha' \alpha) \right) \\
\text{s.t.} & \quad 1'(\alpha^* - \alpha) = 0 \\
& \quad \alpha^* \geq 0, \quad \alpha \geq 0
\end{align*}
\]
The vectors involved in this program are of dimension \( \ell \), with \( 1 = [1 \cdots 1]' \), \( 0 = [0 \cdots 0]' \), \( \varepsilon = [\varepsilon \cdots \varepsilon]' \), \( y = [y_1 \cdots y_\ell]' \) and \( K(Z) \) is the Gram matrix \( K_{Zij} = k_z(z_i, z_j) \). Here, \( \varepsilon \) is the parameter of the Vapnik’s \( \varepsilon \)-insensitive quadratic loss function and \( C \) is a user-defined constant penalizing data points which fail to meet the \( \varepsilon \)-deviation constraint.

In our case, we are interested in learning the mapping from \( \mathcal{F}_x \) to \( \mathbb{R}^d \). In order to learn this mapping, we construct \( d \) (the dimension of input space) SVR machines \( f_1, \ldots, f_d \). Each \( f_i \) is trained to estimate the \( i \)-th coordinate of the vector \( x_t \) given the coordinates vector \( z_i \) of \( x_t \) with respect to \( U \). Denoting by \( z_u \) the function which maps a vector \( x \) to its coordinate vector \( z \) in \( U \), the \( d \) machines provide the mapping \( \psi \):
\[
\psi : \mathcal{F}_x \rightarrow \mathbb{R}^d \\
x \mapsto [f_1(z_u(x)) \cdots f_d(z_u(x))]' \tag{8}
\]
which can be used to estimate the preimages. Using \( \psi \), and noting that the program involved by the SVR algorithm is convex, the estimation of the preimages does not have to deal with any problem of local minima.

### 3 Numerical Results

In this section we present experiments on highly nonlinear time series prediction with Kernel Dynamical Modeling (KDM). As the two series we consider are one dimensional we use the following setup. Each series of length \( T \) is referred to as \( x_{1:T} \). In order to model it, we introduce an embedding dimension \( d \) and a step size \( \kappa \) such that vectors \( x_t = (x_t, x_{t-1}, \ldots, x_{t-(d-1)\kappa})' \) are considered. We make the comparison of our algorithm to the performance achieved by an SVR for nonlinear time series analysis [9, 10], where the mapping associating \( x_t \) to \( x_{t+\kappa} \) is learned. The hyperparameters (kernel parameter and SVR penalization constant \( C \)) are computed with respect to the one-step prediction error measured on a test set, while the value of \( \varepsilon \) is set to 1e-4. Prediction quality is assessed on an independent validation sequence on which root mean squared error (RMSE) is computed.

Two kinds of prediction capacity are evaluated. The first one is a one-step prediction when after a prediction has been made by an algorithm, the true value is used to estimate next time series output. The second one is a multi-steps or trajectory prediction, where the prediction made by the considered model serves as a basis for future ones.

In order to make a prediction for a time \( t > T \), we suppose that we are provided with the vector \( x_{t-1} \), which may have been observed or computed. We determine the coordinates \( z_{t-1} \) of \( x_{t-1} \) with respect to \( U \) and infer the value of \( z_t \) by \( z_t = A_z z_{t-1} + \mu_z \) (see equation (7)); \( \psi \) is then used to recover an estimation of \( x_{t+1} \) (cf. equation (8)). In all our experiments we have made the rude –yet efficient– choice of the linear kernel for \( k_z \).

#### 3.1 Mackey-Glass Time Series Prediction

The Mackey-Glass time series comes from the modelling of blood cells production evolution. It is a one-dimensional signal determined by the following equation
\[
\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\tau)}{1+x(t-\tau)^10}
\]
Table 1: Error (RMSE) of one-step prediction and trajectory prediction for gaussian and polynomial kernels for $MG_{17}$ time series. Regularizing values used for KDM are in subscript. The best results are italicized.

<table>
<thead>
<tr>
<th></th>
<th>gaussian</th>
<th>polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1S</td>
<td>100S</td>
</tr>
<tr>
<td>SVR</td>
<td>0.0812</td>
<td>0.2361</td>
</tr>
<tr>
<td>KDM$_0$</td>
<td>0.0864</td>
<td>0.2906</td>
</tr>
<tr>
<td>KDM$_{0,1}$</td>
<td>0.0863</td>
<td>0.2893</td>
</tr>
<tr>
<td>KDM$_1$</td>
<td>0.0859</td>
<td>0.2871</td>
</tr>
<tr>
<td>KDM$_{10}$</td>
<td>0.0844</td>
<td>0.2140</td>
</tr>
<tr>
<td>KDM$_{100}$</td>
<td>0.0899</td>
<td>0.1733</td>
</tr>
</tbody>
</table>

which, for values of $\tau$ greater than 16.8, shows some highly nonlinear chaotic behavior (see Figure 1 left).

Figure 1: (left) 100 points of the Mackey-Glass time series $MG_{17}$, (right) the first 350 points of the Laser time series.

For our experiments, we focus on $MG_{17}$, the time series associated to the parameter $\tau = 17$. We construct embedding vectors of size $d = 6$ and step size $\kappa = 6$. As $x_t$ is used to predict $x_{t+\kappa}$, the whole dataset can be divided into six “independent” datasets, the first one $S_1$ containing $x_1+(d-1)\kappa$, the second one $S_2$, $x_2+(d-1)\kappa$, . . . , and the sixth one $S_6$, $x_{6\kappa}$. Learning is done as follows. The first 100 points of $S_1$ are used to learning, while the first 100 points of $S_2$ serve to choose the hyperparameters. The prediction error is measured with respect to the points in the range 201 to 300 of $S_1$.

Table 1 reports the RMSE error obtained with gaussian and polynomial kernels, while 1S and 100S respectively stand for one-step prediction and multi-steps prediction over the 100 future observations.

SVR one-step prediction with gaussian kernel gives the best RMSE. None of the tested regularizers allows KDM to perform better, even if the prediction error obtained with them is never more than 10% away from SVR error.

KDM trajectory prediction with gaussian kernel and regularizer $\gamma = 100$ leads to the best error. It is around 17% lower than that of SVR multi-steps prediction while KDM with no regularizer gives the poorest prediction, emphasizing the importance of the regularizer.

Regarding one-step prediction with polynomial kernel, there is no significant difference
Table 2: Error (RMSE) of one-step prediction and trajectory prediction for gaussian and polynomial kernels for Laser time series. Regularizing values used for KDM are in subscript. The best results are italicized.

<table>
<thead>
<tr>
<th></th>
<th>gaussian</th>
<th></th>
<th>polynomial</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1S</td>
<td>100S</td>
<td>1S</td>
<td>100S</td>
</tr>
<tr>
<td>SVR</td>
<td>15.81</td>
<td>67.57</td>
<td>18.14</td>
<td>66.73</td>
</tr>
<tr>
<td>KDM(_0)</td>
<td>67.95</td>
<td>416.2</td>
<td>43.92</td>
<td>68.90</td>
</tr>
<tr>
<td>KDM(_{0.1})</td>
<td>16.59</td>
<td>69.65</td>
<td>22.37</td>
<td>69.60</td>
</tr>
<tr>
<td>KDM(_1)</td>
<td>13.96</td>
<td>70.16</td>
<td>18.13</td>
<td>70.65</td>
</tr>
<tr>
<td>KDM(_{10})</td>
<td>15.18</td>
<td>66.82</td>
<td>17.39</td>
<td>69.43</td>
</tr>
<tr>
<td>KDM(_{100})</td>
<td>18.65</td>
<td>56.53</td>
<td>17.61</td>
<td>53.84</td>
</tr>
</tbody>
</table>

between the performance achieved by SVR and that of KDM, when regularizer is 0, 0.1, 1 or 10. For a regularizer γ = 100, KDM however leads to the best one-step prediction error, around 16% lower than that obtained by SVR prediction.

The dash '-' appearing in the first line of the table means that trajectory prediction made by SVR with a polynomial kernel has failed to give finite values predictions. On the contrary, KDM never shows this kind of behavior. For a regularizer value of γ = 100, it even gives the best trajectory prediction error.

3.2 Laser Time Series Prediction

This time series is the Dataset A from the Santa Fe time series prediction competition. It is a univariate time series from an experiment conducted in a physics laboratory (Figure 1 right) represents the first 350 points of the series). An embedding dimension \( d = 3 \) and a step size \( \kappa = 1 \) are used. The dataset is divided as follows. The first 100 points are used for training, whereas points in the range 201 to 300 provide a test set to select hyperparameters. Validation error (RMSE) is evaluated against points in the range 101 to 200.

Table 2 reports validation error when the described experimental setup is carried out. The most striking information provided by this table is the large error achieved by KDM with no regularizer when a gaussian kernel is used. Looking at the other RMSE values corresponding to different regularizers, the importance of penalizing transition matrix with large entries is underlined.

Besides, when the regularizer γ is appropriately chosen, we see that KDM with gaussian kernel can achieve very good predictions, for one-step prediction and multi-steps prediction as well. KDM one-step best prediction error is however not as far from SVR one-step prediction (about 10% lower) than KDM multi-steps is from its SVR counterpart (around 16% lower).

When a polynomial kernel is used, we observe that KDM with no regularizer provide poor results with regards to one-step prediction error. Unlike to what occurs with the use of a gaussian kernel, KDM with no regularization does not show bad multi-steps prediction ability. Looking at the other entries of this table once again shows that KDM can give very good predictions when a well-suited regularizer is chosen. Hence, we notice that the best multi-steps prediction error of KDM is above 19% better than that obtained by SVR multi-steps prediction.
4 Discussion

4.1 Another Way of Choosing the Parameters

The introduction of a basis $U$ allows to find the parameters of KDM without computing any inversion of infinite dimensional matrices or division by the dimension of $F$. There is, however, a more elegant way to find these parameters when $\sigma^2$ is assumed to be known. In this case, equation (5) needs not to be considered any longer. Considering the prior $p_A(A^\phi) \propto \exp(-\frac{1}{2\sigma^2} \text{trace}(A^\phi'A^\phi))$, for a user defined $\gamma$, the maximum a posteriori for $A^\phi$ is obtained as:

$$A^\phi = \left( \sum_{t=2}^{T} x^\phi_{t-1} x^\phi_{t-1}' - \frac{1}{T-1} \sum_{t=2}^{T} x^\phi_{t} x^\phi_{t}' \right)^{-1} \left( \gamma I + \sum_{t=2}^{T} x^\phi_{t-1} x^\phi_{t-1}' - \frac{1}{T-1} \sum_{t=2}^{T} x^\phi_{t-1} x^\phi_{t-1}' \right)^{-1}.$$

Introducing the matrix $X^\phi = [x^\phi_1 \cdots x^\phi_T]$, the $T$-dimensional vectors $f := [0 \ 1 \cdots 1]'$, $g := [1 \cdots 1 \ 0]'$, the $T \times T$ matrix $P = (P_{ij}) = (\delta_{i,j+1})$ defining $J = P - f g'/(T-1)'$ and $M = \text{diag}(g) - gg'/(T-1)'$, $A^\phi$ can be rewritten as

$$A^\phi = \left( X^\phi J X^\phi' \right) \left( \gamma I + X^\phi M X^\phi' \right)^{-1}$$

$$= \frac{1}{\gamma} X^\phi J \left[ I - \frac{1}{\gamma} K M \left( 1 + \frac{1}{\gamma} M K M \right)^{-1} M \right] X^\phi'.$$

thanks to the Sherman-Woodbury formula, $K$ being the Gram matrix associated to $x^\phi_{1:T}$.

It is thus possible to directly determine the matrix $A^\phi$ when $\sigma^2$ is known, the same holding for $\mu^\phi$ since equation (5) remains unchanged.

4.2 Link to Kalman Filtering

The usual way to recover a noisy nonlinear signal is to use the Extended Kalman Filter (EKF) [15] or the Unscented Kalman Filter (UKF) [5]. The use of these algorithms involves two steps. First, the clean dynamics, as given by $h$ in equation (1) is learned by a regressor, e.g., a multilayer perceptron. Given a noisy time series coming from the same driving process $h$, EKF and UKF then process that series by respectively a first-order linearization of $h$ and an efficient ‘sampling’ method to determine the clean signal. Apart from the latter essential approximations done by these algorithms, the core of EKF and UKF resembles that of classical Kalman filtering (and smoothing).

Regarding the performances of KDM to learn a complex dynamics, it could be directly used to model the process $h$. In addition, its matricial formulation is well suitable for the traditional matrices computations involved by the filtering task (see [6, 11] for details on these computations). This link between KDM and Kalman filtering is the purpose of an ongoing work, and our first insights into the possibility of mixing KDM with Kalman filtering tell us that inverting possibly infinite dimensional matrices can be dealt with easily thanks to the Sherman-Woodbury formula. Hence, tackling the task of denoising highly nonlinear signal seems to be possible without having to resort to any approximated filtering scheme.

5 Conclusion and Future Work

Three main results are presented: first, we introduce KDM, a kernel extension of linear dynamical model and show how the kernel trick allows to learn a corresponding linear model
in the feature space. Second, an original solution based on Support Vector Regression has been applied for the preimage problem. This approach, far to be specific to our algorithm could be applied to any kernel-based method (for instance Kernel-PCA). Third, Kernel Dynamical Model can be linked to Kalman Filter model with a hidden state process living in the feature space.

In the framework of time series prediction, KDM proves to work very well and to compete with the best time series predictors particularly on long time range prediction.

To conclude, this work can lead to several future directions. All classic tasks involving a dynamic setting such as filtering/predicting (e.g. tracking) and smoothing (e.g. time series denoising) can be tackled by our approach and have to be tested. Moreover, the kernel approach could also be applied to linear dynamical model with hidden state and could provide a kernelized version of Kalman Filtering. Lastly, the use of kernel opens the door to dealing with structured data, making KDM a very attractive tool in many areas such as bioinformatics, texts and video application.

References


