ESTIMATION OF THE FORGETTING FACTOR IN KERNEL RECURSIVE LEAST SQUARES

*Steven Van Vaerenbergh, Ignacio Santamaría**

Department of Communications Engineering
University of Cantabria, Spain

Miguel Lázaro-Gredilla

Dept. of Signal Processing and Communications
Universidad Carlos III de Madrid, Spain

ABSTRACT

In a recent work we proposed a kernel recursive least-squares tracker (KRLS-T) algorithm that is capable of tracking in non-stationary environments, thanks to a forgetting mechanism built on a Bayesian framework. In order to guarantee optimal performance its parameters need to be determined, specifically its kernel parameters, regularization and, most importantly in non-stationary environments, its forgetting factor. This is a common difficulty in adaptive filtering techniques and in signal processing algorithms in general. In this paper we demonstrate the equivalence between KRLS-T's recursive tracking solution and Gaussian process (GP) regression with a specific class of spatio-temporal covariance. This result allows to use standard hyperparameter estimation techniques from the Gaussian process framework to determine the parameters of the KRLS-T algorithm. Most notably, it allows to estimate the optimal forgetting factor in a principled manner. We include results on different benchmark data sets that offer interesting new insights.

Index Terms— kernel recursive least squares, Gaussian processes, forgetting factor, adaptive filtering

1. INTRODUCTION

The recursive least-squares (RLS) algorithm is one of the most popular adaptive filters of the past few decades [1, 2]. It allows to retrieve the least-squares linear predictor of a stationary system in an efficient recursive manner. The standard RLS formulation often includes a forgetting factor $\lambda \in (0, 1]$ that allows it to deal with non-stationary systems to some extent, while in order to obtain true tracking capabilities a more general extended recursive least-squares (EX-RLS) algorithm is required [1, 2].

Recently, a class of kernel-based adaptive filter algorithms have emerged [3, 4], following the success of kernel methods such as the support vector machine and kernel principal component analysis [5, 6]. By casting the data into a high-dimensional reproducing kernel Hilbert space, kernel methods allow to solve nonlinear learning problems in the input space as convex optimization problems in the transformed space. We focus on kernel recursive least-squares (KRLS) algorithms, which are kernelized versions of classical RLS algorithms. Standard KRLS algorithms are designed for stationary scenarios only, and they have been successfully applied to signal processing, communications, control and pattern analysis [3, 4]. In order to equip KRLS algorithms with tracking capabilities, additional measures are required. Some interesting approaches include the sliding-window approach from [7] and the kernel-based EX-RLS algorithm from [4].

*This work was supported by MICINN (Spanish Ministry for Science and Innovation) under grants TEC2010-19545-C04-03 (COSIMA) and CONSOLIDER-INGENIO 2010 CSD2008-00010 (COMONSENS).

In [8, 9], a novel KRLS Tracker (KRLS-T) algorithm was devised that explicitly handles uncertainty about the data, based on a probabilistic Bayesian framework. This algorithm incorporates a new forgetting mechanism called "back to the prior" (B2P) that enables it to handle non-stationary scenarios. The KRLS-T algorithm is rooted in Gaussian processes (GP), which are stochastic processes that can be used as a prior over functions within the Bayesian framework. They are often used in the batch regression setting: By assuming that a regression data set has been generated by a latent function plus Gaussian noise and by placing a GP prior over such latent function, it is possible to analytically infer a posterior distribution over latent functions and thus perform predictions. On the other hand, KRLS algorithms are designed to perform regression in an online manner, i.e., when data points are made available on a one-at-a-time basis. After all data points have been observed, a properly implemented KRLS achieves the same solution as a batch GP. This equivalence has been shown in [8, 9, 10, 11]. Indeed, the standard KRLS can be understood as an online GP [8, 9, 10].

This paper is devoted to show the equivalence between the KRLS Tracker with back-to-the-prior forgetting and a standard GP with a specially crafted covariance function. This is relevant for two reasons: First, it proves that back-to-the-prior forgetting is a principled forgetting approach that fits nicely within a global Bayesian model; and second, and practically more relevant, it provides a method to select all the hyperparameters of KRLS, including its forgetting factor, a problem that had not been addressed to date in kernel adaptive filtering literature.

The rest of this paper is structured as follows: In Sections 2 and 3 we provide short reviews of Gaussian process regression and the KRLS-T algorithm, respectively. In Section 4 we link both concepts by demonstrating how KRLS-T is equivalent to Gaussian processes regression with a specific class of spatio-temporal covariance. We then indicate how this equivalence allows the estimation of KRLS-T's forgetting factor and other hyperparameters through standard GP regression. In Section 5 we illustrate the proposed procedure with a set of numerical examples, and we finish by summarizing the main conclusions of this work in Section 6.

2. REVIEW OF GAUSSIAN PROCESS REGRESSION

We begin with a brief review of GP regression. The interested reader is referred to [12] for a thorough treatment.

Assume we are given a set of independent and identically distributed (i.i.d.) samples $\mathcal{D} \equiv \{\mathbf{x}_t, y_t | t = 1, \dots n\}$, where each D-dimensional input $\mathbf{x}_t$ is associated to a scalar output y_t. The regression task goal is, given a new input $\mathbf{x}_*$, to predict the corresponding output y_* based on $\mathcal{D}$.

The GP regression model assumes that the outputs can be mod-

978-1-4673-1026-0/12/$31.00 ©2012 IEEE

eled as some noiseless latent function of the inputs plus an independent noise $y_t = f(\mathbf{x}_t) + \varepsilon_t$, and then sets a zero mean[1] GP prior on $f(\mathbf{x})$ and a Gaussian prior on ε_t:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{x}, \mathbf{x}')), \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma^2),$$

where σ^2 is a hyperparameter that specifies the noise power. The notation $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ refers to a GP distribution over functions in terms of a mean function $m(\mathbf{x})$ (zero in this case) and a covariance function $k(\mathbf{x}, \mathbf{x}')$. The covariance function specifies the a priori relationship between values $f(\mathbf{x})$ and $f(\mathbf{x}')$ in terms of their respective locations, and it is parameterized by a small set of hyperparameters $\boldsymbol{\theta}$.

By definition, the marginal distribution of a GP at a finite set of points is a joint Gaussian distribution, with its mean and covariance being specified by the homonymous functions evaluated at those points. Thus, the joint distribution of outputs $\mathbf{y} = [y_1, \ldots, y_n]^\top$ and the corresponding latent vector $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^\top$ is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2\mathbf{I} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{bmatrix}\right).$$

$\mathbf{K}$ is a matrix with elements $k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{I}$ is used to denote the identity matrix of the appropriate size. By conditioning on the observed outputs $\mathbf{y}$, the posterior over the latent vector can be inferred

$$\begin{aligned} p(\mathbf{f}|\mathbf{y}) &= \mathcal{N}(\mathbf{f}|\mathbf{K}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, \mathbf{K} - \mathbf{K}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{K}) \\ &= \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \end{aligned} \tag{1}$$

being the most expensive operation the inversion of the $n \times n$ matrix $\mathbf{K} + \sigma^2\mathbf{I}_n$, which is computable in $\mathcal{O}(n^3)$ time. This posterior can also be computed incrementally in an online manner as samples are made available. In that case, the cost of adding sample t is $\mathcal{O}(t^2)$, resulting in the same overall cost.

If the posterior over the latent function given the observations is known to be $\mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the predictive distribution of a new output y_* at location $\mathbf{x}_*$ can be computed as

$$\begin{aligned} p(y_*|\mathbf{y}) &= \int p(y_*|f_*)p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})\mathrm{d}\mathbf{f}\mathrm{d}f_* \\ &= \int \mathcal{N}(y_*|f_*, \sigma^2)\mathcal{N}(f_*|\mathbf{q}^\top\mathbf{f}, \gamma^2)\mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\mathrm{d}\mathbf{f}\mathrm{d}f_* \\ &= \mathcal{N}(y_*|\mathbf{q}^\top\boldsymbol{\mu}, \sigma^2 + \gamma^2 + \mathbf{q}^\top\boldsymbol{\Sigma}\mathbf{q}), \end{aligned} \tag{2}$$

where we have defined $\mathbf{q} = \mathbf{K}^{-1}\mathbf{k}_*$ and $\gamma^2 = k_{**} - \mathbf{k}_*^\top\mathbf{K}^{-1}\mathbf{k}_*^\top$; $\mathbf{k}_*$ is a vector with elements $k(\mathbf{x}_i, \mathbf{x}_*)$ and $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$. Eq. (2) provides the test predictive distribution given *any* posterior over $\mathbf{f}$. If we plug in the posterior obtained in (1), we get the familiar equations

$$p(y_*|\mathbf{x}_*, \mathbf{y}) = \mathcal{N}(y_*|\mu_{\text{GP}*}, \sigma^2_{\text{GP}*}) \tag{3a}$$

$$\mu_{\text{GP}*} = \mathbf{k}_*^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y} \tag{3b}$$

$$\sigma^2_{\text{GP}*} = \sigma^2 + k_{**} - \mathbf{k}_*^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_*. \tag{3c}$$

[1] It is customary to subtract the sample mean to data $\{y_t\}_{t=1}^n$, and then to assume a zero mean model.

Algorithm 1 Kernel Recursive Least-Squares Tracker

Parameters: Forgetting factor λ, regularization, kernel function $k(\mathbf{x}, \mathbf{x}')$ including its parameters, and budget M.
Observe $(\mathbf{x}_1, y_1)$.
Initialize $\boldsymbol{\mu}_1$, $\boldsymbol{\Sigma}_1$, $\mathbf{Q}_1$.
for $t = 1, 2, \ldots$ **do**
  Forget: update $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$.
  Observe new input $\mathbf{x}_{t+1}$
  Calculate predictive mean $\hat{y}_{t+1}$.
  Calculate predictive variance $\hat{\sigma}^2_{yt+1}$.
  Observe actual output y_{t+1}.
  Compute $\boldsymbol{\mu}_{t+1}$, $\boldsymbol{\Sigma}_{t+1}$, $\mathbf{Q}_{t+1}$.
  Add basis $\mathbf{x}_{t+1}$ to the dictionary.
  if Number of bases in the dictionary $> M$ **then**
    Determine the least relevant basis $\mathbf{x}_i$.
    Remove basis $\mathbf{x}_i$ from $\boldsymbol{\mu}_{t+1}$, $\boldsymbol{\Sigma}_{t+1}$, $\mathbf{Q}_{t+1}$.
    Remove basis $\mathbf{x}_i$ from the dictionary.
  end if
end for

3. KERNEL RECURSIVE LEAST-SQUARES TRACKER

Standard KRLS focusses on obtaining the solution (3b) in an online recursive manner. Its main problem consists in limiting the growth of the functional representation during online operation, for which it typically builds a compact dictionary of relevant bases (see [3]).

Though standard KRLS is an online algorithm, it assumes a stationary scenario, i.e. the latent function does not change over time. In a non-stationary scenario, however, the algorithm should *track* the changes of the latent function. This is possible by weighting past data less heavily than more recent data, i.e. it should *forget* past observations. Unlike linear RLS, the standard KRLS formulation does not allow to include such a forgetting mechanism directly. Nevertheless, in [8, 9] it was shown that tracking capabilities can be included into KRLS by explicitly modeling the algorithm's uncertainty about the data, which becomes possible by adopting a Bayesian framework. A summary of the resulting KRLS-T algorithm can be found in Alg. 1. In the sequel we highlight its main characteristics. For a more detailed description refer to [9].

3.1. Algorithm variables

KRLS-T stores and updates three variables: 1) the current predictive mean values $\boldsymbol{\mu}_t$ corresponding to the bases in its dictionary; 2) the corresponding inverse kernel matrix $\mathbf{Q}_t = \mathbf{K}_t^{-1}$, which represents the prior, and 3) the corresponding predictive variance $\boldsymbol{\Sigma}_t$, which captures the covariance of the posterior.

3.2. Forgetting mechanism

KRLS-T includes a forgetting mechanism that admits different forms of forgetting. We focus on back-to-the-prior forgetting, in which the mean and covariance are updated through

$$\boldsymbol{\mu} \leftarrow \sqrt{\lambda}\boldsymbol{\mu} \tag{4a}$$

$$\boldsymbol{\Sigma} \leftarrow \lambda\boldsymbol{\Sigma} + (1 - \lambda)\mathbf{K}. \tag{4b}$$

As shown in [9], this particular form of forgetting corresponds to blending the informative posterior with a "noise" distribution that uses the same color as the prior. In other words, forgetting occurs by taking a step back towards the prior knowledge. Since the prior

has zero mean, the mean is simply scaled by the square root of the forgetting factor λ. The covariance, which represents the posterior uncertainty on the data, is pulled towards the covariance of the prior.

If the forgetting step is omitted, the recursively constructed solution is equivalent to the standard GP solution in the batch setting. By introducing forgetting, though, the KRLS-T algorithm may deviate from the purely Bayesian setting, depending on the type of forgetting. In Section 4 we will show that this is not the case for back-to-the-prior forgetting.

3.3. Fixed budget through pruning

Finally, in order to deal with very large data sequences, KRLS-T also includes a pruning mechanism that allows it to maintain its dictionary size M low or even fixed. The latter is achieved by accepting every basis into the dictionary (as long as this does not render $\mathbf{Q}_t$ rank-deficient), and by pruning the least relevant basis at the end of each iteration, once its information is projected onto the remaining bases. Note that this procedure induces a pruning error, rendering the obtained solution an approximation of the optimal solution.

4. GAUSSIAN PROCESS REGRESSION WITH SPATIO-TEMPORAL COVARIANCE

4.1. A non-stationary GP regression model

The standard GP framework that has been described in Section 2 is designed to deal with *stationary* regression: All observed data points are assumed to have been generated following the same input-output mapping. However, as mentioned in Section 3, it is of high practical interest to be able to deal with non-stationary regression, i.e., to assume that there is some kind of smooth drift in the mapping function, so that it evolves smoothly over time.

Fortunately, it is very simple to use the GP framework from Section 2 to deal with non-stationarity, by extending the latent GP to be a function both of *space* and *time*. This can be achieved by augmenting the input domain with an additional temporal dimension. Specifically, each input sample is extended as $\tilde{\mathbf{x}}_t = [t,\ \mathbf{x}_t^\top]^\top$, where an additional time stamp has been included. For notational simplicity we restrict this discussion to evenly spaced time intervals, which is the most common case. Nevertheless, the GP can seamlessly deal with uneven sampling, and also with several samples having the same time stamp.

In order to apply the GP framework on this augmented input domain, we need to define a spatio-temporal covariance function. We will consider covariance functions of the type

$$\begin{aligned} k_{st}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') &= k_t(t, t') k_s(\mathbf{x}, \mathbf{x}') \\ &= \lambda^{\frac{|t-t'|}{2}} k_s(\mathbf{x}, \mathbf{x}') \qquad \lambda \in (0, 1], \end{aligned} \tag{5}$$

which are defined as the product of an Ornstein-Uhlenbeck temporal covariance function with any valid spatial covariance function $k_s(\cdot, \cdot)$. This defines an autoregressive (AR) process of order 1 over time. We will refer to the covariance model in (5) as "spatio-temporal AR1" (STAR1) covariance.

All previous equations from the standard GP framework apply, as long as the proper time stamp is added to each sample. In particular, if all samples have the same time stamp, $k_{st}(\cdot, \cdot)$ reduces to $k_s(\cdot, \cdot)$ and time plays no role, so it does not need to be considered. Also, when making predictions, one needs to know not only *where* a prediction must be made, but also *when*. This is typically one or more time-steps ahead into the future, but predictions about the past are also possible, for instance when data smoothing is required.

4.2. Equivalence between KRLS-T and spatio-temporal GP regression

The use of an AR(1) process over time results in interesting independence properties. Let us have a look at the evolution of the posterior over time. We will represent the latent vector at some set of locations at time t as $\mathbf{f}_t = [f([t,\ \mathbf{x}_1^\top]^\top), \ldots, f([t,\ \mathbf{x}_n^\top]^\top)]^\top$. According to the spatio-temporal covariance function, the joint distribution of the latent vector at several consecutive time instants is

$$[\mathbf{f}_1^\top,\ \ldots,\ \mathbf{f}_t^\top,\ \mathbf{f}_{t+1}^\top]^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}_{t+1} \otimes \mathbf{K}),$$

where $\otimes$ is the Kronecker product and $\mathbf{\Lambda}_T$ is a $T \times T$ Toeplitz matrix with $\lambda^{\frac{|j|}{2}}$ on the j-th diagonal.

From the joint, we can obtain conditional $\mathbf{f}_{t+1}|\mathbf{f}_1,\ \ldots,\ \mathbf{f}_t$. Interestingly, due to the special structure of $\mathbf{\Lambda}_{t+1}$, whose inverse is tridiagonal, a term $[\lambda^{\frac{1}{2}},\ \lambda^{\frac{2}{2}},\ \ldots,\ \lambda^{\frac{t}{2}}]\mathbf{\Lambda}_t = [\lambda^{\frac{1}{2}},\ 0,\ \ldots,\ 0]$ appears in the conditional, canceling the contribution from all past instants except the immediately previous, so that the previous conditional equals $\mathbf{f}_{t+1}|\mathbf{f}_t$. In other words, due to the Markov property of the AR(1) process, given the present value of the latent vector, $\mathbf{f}_t$, we know that $\mathbf{f}_{t+1}$ is independent of all past values.

The joint distribution of the latent vector at two consecutive time instants is

$$\begin{bmatrix} \mathbf{f}_t \\ \mathbf{f}_{t+1} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}_2 \otimes \mathbf{K}) = \mathcal{N}\left(\mathbf{0},\ \begin{bmatrix} \mathbf{K} & \sqrt{\lambda}\mathbf{K} \\ \sqrt{\lambda}\mathbf{K} & \mathbf{K} \end{bmatrix}\right).$$

The conditional distribution $p(\mathbf{f}_{t+1}|\mathbf{f}_t)$ is then

$$p(\mathbf{f}_{t+1}|\mathbf{f}_t) = \mathcal{N}(\mathbf{f}_{t+1}|\sqrt{\lambda}\mathbf{f}_t, (1-\lambda)\mathbf{K}),$$

which means that, given the posterior distribution at time t, $p(\mathbf{f}_t|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{\Sigma}_t)$, the posterior at time $t+1$ is

$$\begin{aligned} p(\mathbf{f}_{t+1}|\mathbf{y}) &= \int p(\mathbf{f}_{t+1}|\mathbf{f}_1, \ldots, \mathbf{f}_t) p(\mathbf{f}_1, \ldots, \mathbf{f}_t|\mathbf{y}) \mathrm{d}\mathbf{f}_t \\ &= \int p(\mathbf{f}_{t+1}|\mathbf{f}_t) p(\mathbf{f}_t|\mathbf{y}) \mathrm{d}\mathbf{f}_t \\ &= \int \mathcal{N}(\mathbf{f}_{t+1}|\sqrt{\lambda}\mathbf{f}_t, (1-\lambda)\mathbf{K}) \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{\Sigma}_t) \mathrm{d}\mathbf{f}_t \\ &= \mathcal{N}(\mathbf{f}_{t+1}|\sqrt{\lambda}\boldsymbol{\mu}_t, \lambda\mathbf{\Sigma}_t + (1-\lambda)\mathbf{K}), \end{aligned} \tag{6}$$

which corresponds exactly with the back-to-the-prior forgetting rule of KRLS-T of Eq. (4), as introduced in [8] and further developed in [9]. Thus KRLS-T (with unlimited budget M) is exactly equivalent to the described non-stationary GP. During its learning process, KRLS-T updates its posterior with the corresponding observation at each time step (this step does not require any temporal augmentation, since all samples have the same time stamp and it would have no effect), and then propagates the posterior to the next time step using (6) (i.e, it forgets). With the new posterior, it is capable of making predictions at the new time step (which again requires no temporal augmentation due to the previously mentioned reason). Then the cycle is repeated.

Unlike the non-stationary GP, KRLS-T cannot provide predictions for past time instants, since only the small part of the posterior that is relevant for the current time instant is kept. Fortunately, predictions about the past are not generally required and computational cost is greatly reduced.

4.3. Selecting the forgetting factor and other hyperparameters.

So far we have considered hyperparameters $\{\boldsymbol{\theta}, \sigma, \lambda\}$ as known. This is rarely the case in practice, but often the assumption found in the signal processing literature. Even linear algorithms which do not need kernel hyperparameters or noise power to be selected (RLS, EX-RLS) require a forgetting factor to be specified. The forgetting factor λ tells us whether we are assuming stationarity in data ($\lambda = 1$) or not and to which degree ($0 < \lambda < 1$). Instead of fixing it a priori, it is possible to let the data speak for itself. The GP perspective brings us a principled method for hyperparameter selection. We can marginalize out the latent function and express the probability of the observed data given the hyperparameters as

$$\log p(\mathbf{y}|\boldsymbol{\theta}, \sigma, \lambda) = -\frac{1}{2}\mathbf{y}^\top \left(\boldsymbol{\Lambda} \circ \mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y} - \frac{1}{2}|\boldsymbol{\Lambda} \circ \mathbf{K} + \sigma^2\mathbf{I}| - \frac{n}{2}\log(2\pi), \quad (7)$$

where $\circ$ represents the Hadamard (element-wise) product, and then maximize (7) w.r.t. the hyperparameters, which corresponds to type-II maximum likelihood (ML-II). A fully Bayesian approach would correspond to placing a hyperprior on these hyperparameters to avoid overfitting, but since the posterior is highly peaked around its mode, ML-II yields a good approximation in a computationally efficient way. Gradients can be computed analytically:

$$\frac{\partial \log p(\mathbf{y}|\boldsymbol{\theta}, \sigma, \lambda)}{\partial \boldsymbol{\theta}} = \frac{\text{tr}\left[\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \left(\boldsymbol{\Lambda} \circ \mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\right)\left(\boldsymbol{\Lambda} \circ \frac{\partial \mathbf{K}}{\partial \boldsymbol{\theta}}\right)\right]}{2}$$

$$\frac{\partial \log p(\mathbf{y}|\boldsymbol{\theta}, \sigma, \lambda)}{\partial \boldsymbol{\sigma}} = \text{tr}\left[\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \left(\boldsymbol{\Lambda} \circ \mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\right]\sigma$$

$$\frac{\partial \log p(\mathbf{y}|\boldsymbol{\theta}, \sigma, \lambda)}{\partial \lambda} = \frac{1}{2}\text{tr}\left[\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \left(\boldsymbol{\Lambda} \circ \mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\right)\left(\mathbf{K} \circ \boldsymbol{\Lambda}'\right)\right]$$

with $\boldsymbol{\alpha} = \left(\boldsymbol{\Lambda} \circ \mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y}$ and $\boldsymbol{\Lambda}'$ being an $n \times n$ Toeplitz matrix with $\frac{|j|}{2}\lambda^{\frac{|j|}{2}-1}$ in the j-th diagonal. Therefore, conjugate gradient ascent can be used for maximization as long as the spatial kernel $\mathbf{K}$ is derivable w.r.t. its hyperparameters, which is almost always the case.

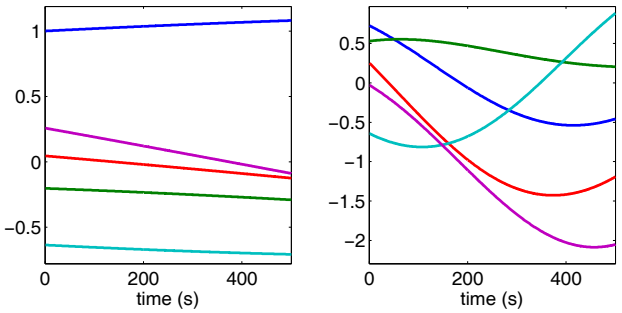
Thus, the proposed procedure can be summarized as follows:

1. Cast the online adaptive regression problem as GP regression with a STAR1 spatio-temporal covariance;
2. Choose the values for $\{\boldsymbol{\theta}, \sigma, \lambda\}$ that maximize (7) on an initial, separate data set;
3. And finally use those values to keep learning and adapting online with KRLS-T.

5. COMPUTER SIMULATIONS

We experimentally demonstrate the proposed procedure for selecting KRLS-T's parameters. In order to infer the hyperparameters we use the GPML toolbox [13]. Matlab code for the KRLS-T algorithm is available at `http://www.tsc.uc3m.es/~miguel`.

Note that KRLS-T uses an additional parameter M that is to be determined according to the available computational budget. We choose this parameter sufficiently high in each experiment so as to minimize the pruning error associated with it.



(a) $f_d = 100$Hz (b) $f_d = 1000$Hz

Fig. 1. Rayleigh channel coefficients (real parts only) for Doppler frequencies of $f_d = 100$Hz and $f_d = 1000$Hz, corresponding to the normalized Doppler frequencies $f_d T = 10^{-4}$ and $f_d T = 10^{-3}$.

5.1. Tracking a Rayleigh fading channel

The first experiment is based on the Rayleigh fading channel tracking scenario from [2, chapter 12]. Rayleigh fading is a mathematical model used to characterize the fading properties of the channel coefficients in wireless communications. The number of paths is chosen as $L = 5$ and the sampling rate is set as $T = 1\mu$s. We consider different scenarios with different Doppler frequencies, ranging from $f_d = 100$Hz, which represents a slowly fading channel, up to $f_d = 10$kHz, representing a fast time-varying channel. In general, a communications channel is considered to be fast time-varying when the normalized Doppler frequency satisfies $f_d T \geq 0.001$, which corresponds to $f_d \geq 1$kHz for the given symbol period T. A representative sample of two typical channel fadings is shown in Fig. 1. Note that we will only consider the real parts of the channel responses in this experiment.

The input signal to the channel consists of 500 Gaussian i.i.d. samples, $x_t \in \mathcal{N}(0, 1)$, and the output y_t is corrupted with 30dB of additive white Gaussian noise. Since we assume the channel order to be known[2], i.e. $L = 5$, the time-embedded input vector to the algorithm is defined as $\mathbf{x}_t = [x_t, x_{t-1}, \ldots, x_{t-4}]^T$. The tracking problem then consists in predicting the output y_{t+1} given the input $\mathbf{x}_{t+1}$ and the data available up to time t, $\mathcal{D}_t$. The experiment is repeated for each chosen Doppler frequency.

Since this is a linear adaptive filtering scenario, the spatial covariance in Eq. (5) is chosen as the linear kernel $k_s(\mathbf{x}, \mathbf{x}') = \mathbf{x}\mathbf{x}'^T$. For each Doppler frequency we estimate the forgetting factor by maximizing the log likelihood (7) of the available training data. The results, averaged out over 25 simulations, are shown in Fig. 2 (top).

We then convert this scenario into a nonlinear tracking problem by applying a saturation nonlinearity $y' = \tanh(y)$ onto the channel output before summing the additive noise. We repeat the estimation of the forgetting factor for this scenario, where we now use a Gaussian kernel as the spatial covariance in (5). The results are shown in Fig. 2 (bottom). Interestingly, lower λ-values are required in the nonlinear scenario. This indicates that forgetting needs to occur faster, which seems reasonable as this is a harder tracking problem.

5.2. Tracking a channel with abrupt changes

In the second experiment we investigate the tracking of a switching channel. Here, a communications channel is to be identified or

[2]If the channel order is not known it should be overestimated, as GP regression allows to suppress any redundant input data components.

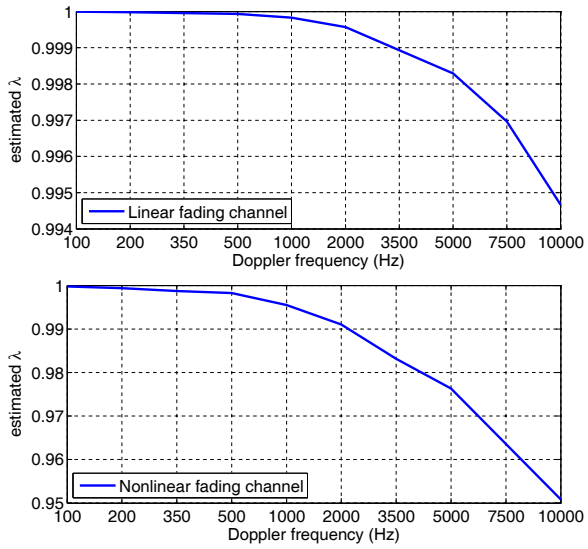


Fig. 2. Estimated forgetting factors for tracking the linear (top) and nonlinear (bottom) fading channels used in experiment 1.

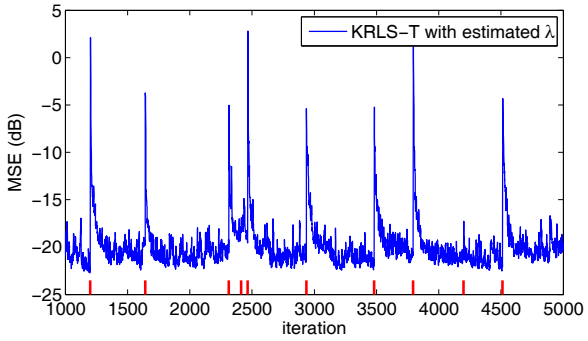


Fig. 3. Tracking results for KRLS-T on the switching channel from experiment 2, using the estimated forgetting factor. Channel switches are indicated as the vertical red lines at the bottom.

equalized, similar to the previous experiment, but instead of changing slowly it shows sudden abrupt changes in its impulse response. While the STAR1 covariance (5) assumes gradual temporal changes and hence does not adjust too well to this model, we can still apply the proposed procedure in order to determine the optimal forgetting factor in case KRLS-T is to be applied as a tracker.

The setup is as follows: At different times throughout the experiment the channel is switched abruptly to a different channel. All channel impulse responses consist of 5 randomly chosen taps. The switch times are controlled by a Poisson process with arrival rate λ_{pp}. Note that the Poisson process arrival rate λ_{pp} is not to be confused with the forgetting factor λ central in this work. The tracking experiment consists in predicting y_{t+1} given the time-embedded input data $\mathbf{x}_{t+1}$ and all previous data $\mathcal{D}_t$.

In a first setup we feed 5000 data points $x_t \in \mathcal{N}(0,1)$ into the system and add white Gaussian noise to its output, with an SNR of 20dB. The first 1000 points are used to determine the hyperparameters, and the remaining 4000 points are used for tracking with KRLS-T. We fix the arrival rate of the Poisson process at $\lambda_{pp} = 0.003$, resulting in 5 channel switches during the training period and 10

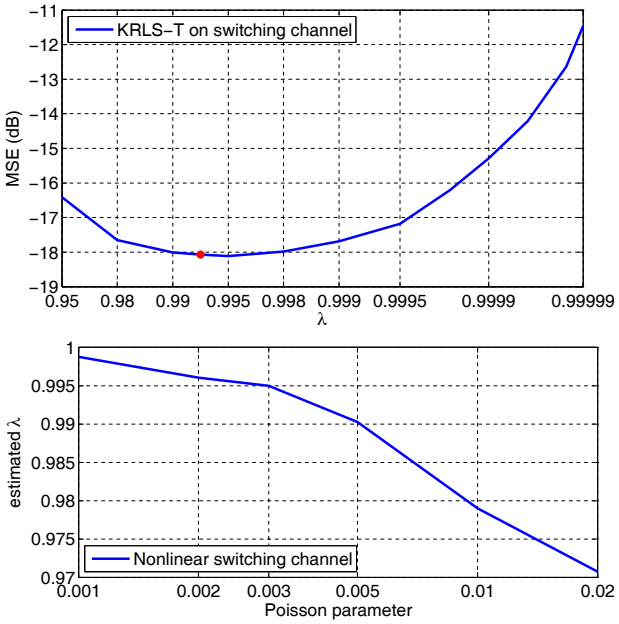


Fig. 4. Top: MSE for KRLS-T with different forgetting factors. The dot marks the estimated forgetting factor. Bottom: Estimated forgetting factor for different arrival rates.

channel switches during the rest of the experiment, in this particular realization. For the spatial covariance we use a standard Gaussian kernel. After maximizing the log likelihood (7) the forgetting factor $\lambda = 0.9918$ is obtained. With this choice of λ we perform tracking with KRLS-T on the next 4000 data. At each time step we measure the MSE on a separate test set of 500 data that are generated with the current channel response. The obtained MSE values are shown in Fig. 3. The average MSE over the entire test data set is -18.06dB.

In order to verify the quality of the forgetting factor estimate, we then run KRLS-T with different values of λ on the same 4000 data. The result is shown in Fig. 4 (top). Observe that the forgetting factor estimated by the proposed method yields an MSE value very close to the minimum, which is found at -18.11dB for $\lambda = 0.995$. The slight difference between the optimal value and the estimated value is mainly due to the statistical differences between the training set and the test set. This difference will disappear as more data are used for training and testing.

Finally we calculate the estimated forgetting factors for different values of the Poisson process arrival rate. The results, averaged out over 15 Monte-Carlo simulations, are shown in Fig. 4 (bottom).

5.3. Prediction of chaotic time series

In the last experiment we apply the proposed procedure to the prediction of two popular time series. The first series is the MG30 Mackey-Glass benchmark, obtained as $dx_t/dt = \beta x_{t-\tau}/(1 + x_{t-\tau}^n) - \gamma x_t$, in which we set $\beta = 0.2$, $\gamma = 0.1$, $n = 10$ and $\tau = 30$. The second series is the first component of the Lorenz attractor, whose components are defined as $dy/dt = \sigma(z - y)$, $dx/dt = \sigma(y - x)$, and $dz/dt = -\beta x + yz$. We set $\beta = 8/3$, $\sigma = 10$, $\rho = 28$, as in [4].

A common question in time-series prediction is how to choose an adequate time-embedding. For time series generated by a deterministic process, a principled tool to find the optimal embedding is Takens' theorem [14]. In case of the MG30 time series, Takens' the-

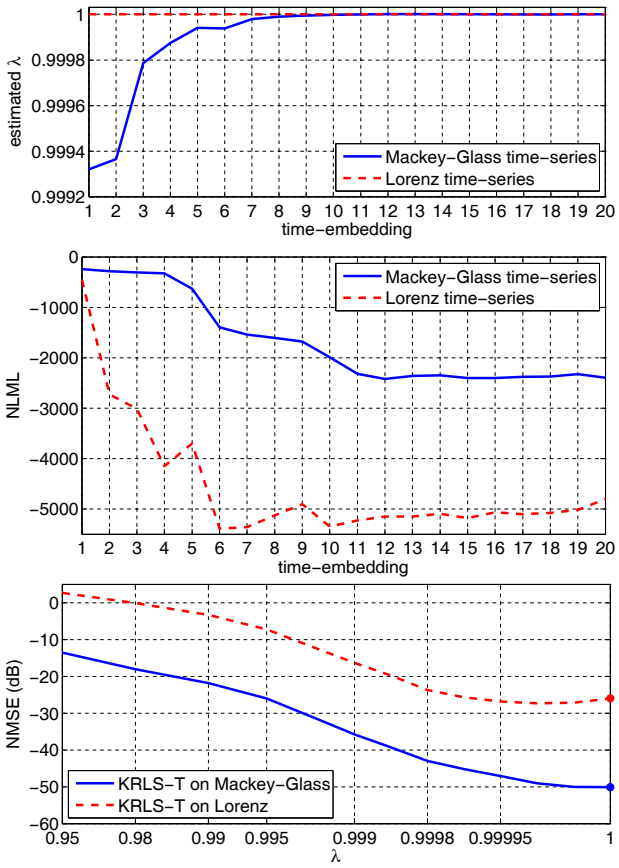


Fig. 5. Results on chaotic time-series prediction.

orem indicates that a good reconstruction of the time series can be achieved with an embedding around 7. For the Lorenz attractor data considered, the embedding should be 5 or higher.

We use the first 500 data of each time series to estimate the hyperparameters. The obtained λ-values are shown in Fig. 5 (top). Since both time series are deterministic, the optimal forgetting factor is found as 1 when sufficient temporal embedding is available. For the MG30 series it is slightly below 1 when the embedding is insufficient, which suggests that it is actually beneficial to forget a small amount of data during each iteration in this case.

In order to select an adequate value of the time-embedding we plot the negative value of the log marginal likelihood (NLML) obtained after optimizing Eq. (7). The results, shown in Fig. 5 (middle) confirm Takens' theorem: The MG30 series can be explained reasonably well with an embedding of 6, while optimal results require embeddings of 11 or higher. The Lorenz series obtains best results for an embedding of 6 or higher. Finally, Fig. 5 (bottom) displays the results for KRLS-T on both series. Embeddings of 12 and 10 were used, respectively, and the estimated λ are marked with a dot.

6. CONCLUSIONS

We have presented a Gaussian process regression model, based on a specific spatio-temporal covariance, that is equivalent to the recently introduced kernel recursive least-squares tracker algorithm. This equivalence proves that the back-to-the-prior forgetting mechanism of KRLS-T fits within a principled Bayesian framework. As a result, it becomes possible to determine all parameters of KRLS-T by applying standard GP hyperparameter estimation. In particular, it allows to determine the forgetting factor, which is an important problem in tracking almost any non-stationary scenario.

The proposed technique has several interesting applications. In this work we conducted experiments in which it was applied to determine the optimal parameters for tracking different types of fading and switching communication channels. We also applied it to the online prediction of chaotic time series, where it allowed to empirically confirm the optimal time-embedding indicated by Takens' theorem.

7. REFERENCES

[1] Simon Haykin, *Adaptive Filter Theory (4th Edition)*, Prentice Hall, Sept. 2001.

[2] Ali H. Sayed, *Fundamentals of adaptive filtering*, Wiley-IEEE Press, 2003.

[3] Yaakov Engel, Shie Mannor, and Ron Meir, "The kernel recursive least squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[4] Weifeng Liu, José C. Príncipe, and Simon Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, Wiley, 2010.

[5] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[6] Bernhard Schölkopf and Alexander J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, Dec. 2001.

[7] Steven Van Vaerenbergh, Javier Vía, and Ignacio Santamaría, "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing Proceedings (ICASSP 2006)*, Toulouse, France, May 2006.

[8] Miguel Lázaro-Gredilla, Steven Van Vaerenbergh, and Ignacio Santamaría, "A Bayesian approach to tracking with kernel recursive least-squares," in *2011 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Sept. 2011, pp. 1 –6.

[9] Steven Van Vaerenbergh, Miguel Lázaro-Gredilla, and Ignacio Santamaría, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1313–1326, Aug. 2012.

[10] Lehel Csató and Manfred Opper, "Sparse online Gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.

[11] Weifeng Liu, Il Park, and José C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1950–1961, 2009.

[12] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[13] Carl Edward Rasmussen and Hannes Nickisch, "Gaussian processes for machine learning (GPML) toolbox," *The Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, 2010.

[14] Floris Takens, "Detecting strange attractors in turbulence," *Dynamical Systems and Turbulence*, vol. 898, pp. 366–381, 1981.