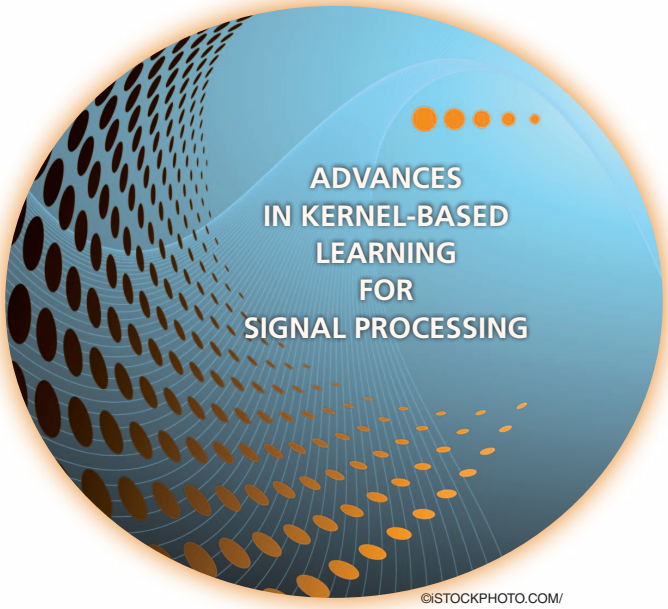
Fernando Pérez-Cruz, Steven Van Vaerenbergh, Juan José Murillo-Fuentes,
Miguel Lázaro-Gredilla, and Ignacio Santamaría

Gaussian Processes for Nonlinear Signal Processing



An overview of recent advances

Gaussian processes (GPs) are versatile tools that have been successfully employed to solve nonlinear estimation problems in machine learning but are rarely used in signal processing. In this tutorial, we present GPs for regression as a natural nonlinear extension to optimal Wiener filtering. After establishing their basic formulation, we discuss several important aspects and extensions, including recursive and adaptive algorithms for dealing with nonstationarity, low-complexity solutions, non-Gaussian noise models, and classification scenarios. Furthermore, we provide a selection of relevant applications to wireless digital communications.

INTRODUCTION

GPs are Bayesian state-of-the-art tools for discriminative machine learning, i.e., regression [1], classification [2], and dimensionality reduction [3]. GPs were first proposed in statistics by Tony O'Hagan [4] and they are well known to the geostatistics community as *kriging*. However, due to their high computational complexity, they did not become widely applied tools in machine learning until the early 21st century [5]. GPs can be interpreted as a family of kernel methods with the additional advantage of providing a full conditional statistical description for the predicted variable, which can be primarily used to establish confidence intervals and to set hyperparameters. In a nutshell, GPs assume that a GP prior governs the set of possible latent functions (which are unobserved) and the likelihood (of the latent function) and observations shape this prior to produce posterior probabilistic estimates. Consequently, the joint distribution of training and test data is a multidimensional Gaussian and the predicted distribution is estimated by conditioning on the training data.

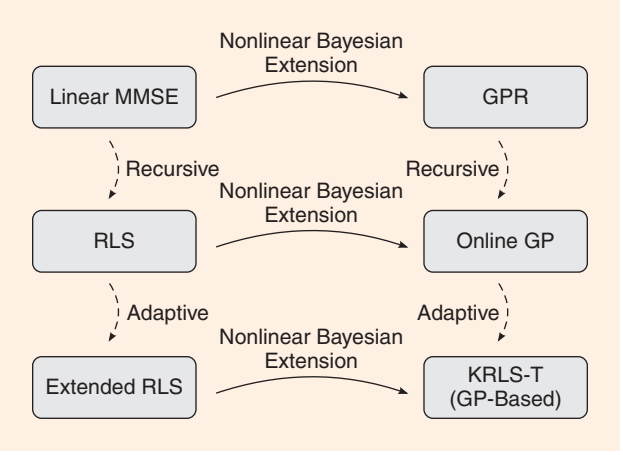
While GPs are well-established tools in machine learning, they are not as widely used by the signal processing community as are neural networks or support vector machines (SVMs). There are several explanations for the limited use of GPs in signal processing problems. First, they do not have a simple intuition for classification problems. Second, their direct implementation is computationally demanding. Third, their plain vanilla version might seem uptight and not flexible enough. Fourth, to most signal processing experts GP merely stands for a noise model and not for a flexible algorithm that they should be using.

Digital Object Identifier 10.1109/MSP.2013.2250352
Date of publication: 12 June 2013

1053-5888/13/$31.00©2013IEEE

GPs FOR MACHINE LEARNING

In this article, we present an overview on GPs explained for and by signal processing practitioners. We introduce GPs as the natural nonlinear Bayesian extension to the linear minimum mean square error (MMSE) and Wiener filtering, which are central to many signal processing algorithms and applications. We have summarized in Figure 1 the relationship between the regression techniques introduced throughout the article. We believe that GPs provide the correct approach to solve an MMSE filter nonlinearly, because they naturally extend least squares to nonlinear solutions through the kernel trick; they use a simple yet flexible prior to control the nonlinearity; and evidence sampling or maximization allows setting the hyperparameters without overfitting. This last feature is most interesting: by avoiding cross-validation we are able to optimize over a larger number of hyperparameters, thus increasing the available kernel expressiveness. Additionally, GP provides a full statistical description of its predictions.



[FIG1] The relationship between the regression techniques discussed in this tutorial.

MINIMUM MEAN SQUARE ERROR: A STARTING POINT

GPs can be introduced in a number of ways and we, as signal processing practitioners, find it particularly appealing to start from the MMSE solution. This is because the Wiener solution, which is obtained by minimizing the MSE criterion, is our first approach to most estimation problems and, as we show, GPs are its natural Bayesian extension.

GPs ARE VERSATILE TOOLS THAT HAVE BEEN SUCCESSFULLY EMPLOYED TO SOLVE NONLINEAR ESTIMATION PROBLEMS.

Many signal processing problems reduce to estimating from an observed random process $\mathbf{x} \in \mathbb{R}^p$ another related process $y \in \mathbb{R}$. These two processes are related by a probabilistic, possibly unknown, model $p(\mathbf{x}|y)$. It is well known that the unconstrained MMSE estimate,

$$\arg\min_{f(\mathbf{x})} E[\| y - f(\mathbf{x})^2 \|], \tag{1}$$

coincides with the conditional mean estimate of y given $\mathbf{x}$

$$f_{\text{mmse}}(\mathbf{x}) = E[y|\mathbf{x}] = \int y p(y|\mathbf{x}) dy = \int y \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} dy. \tag{2}$$

If $p(y, \mathbf{x})$ is jointly Gaussian, i.e., $p(y)$ and $p(\mathbf{x}|y)$ are Gaussians and $E[\mathbf{x}|y]$ is linear in y, this solution is linear. If y and $\mathbf{x}$ are zero mean, the solution yields $E[y|\mathbf{x}] = \mathbf{w}^\top \mathbf{x}$, where

$$\mathbf{w}_{\text{mmse}} = \arg\min_{\mathbf{w}} E[(y - \mathbf{w}^\top \mathbf{x})^2] = (E[\mathbf{x}\mathbf{x}^\top])^{-1} E[\mathbf{x}\, y]. \tag{3}$$

Furthermore, these expectations can be easily estimated, using the sample mean, from independently and identically distributed (i.i.d.) samples drawn from $p(\mathbf{x}|y)$ and $p(y)$, particularly $\mathcal{D}_n = \{\mathbf{x}_i, y_i\}_{i=1}^n$.

However, if $\mathbf{x}$ is not linearly related to y (plus Gaussian noise) or y is not Gaussian distributed, the conditional estimate of y given $\mathbf{x}$ is no longer linear. Computing the nonlinear conditional mean estimate in (2) directly from $\mathcal{D}_n$ either leads to overfitted solutions, because there are no convergence guarantees for general density estimation [6], or to suboptimal solutions, if we restrict the density model to come from a narrow class of distributions. For instance, in channel equalization, although suboptimal, the sampled version of (3) is used due to its simplicity. One viable solution would be to minimize the sampled version of (1) with a restricted family of approximating functions to avoid overfitting. Kernel least squares (KLS) [7] and GPs for regression (GPR), among others, follow such an approach.

GPs FOR REGRESSION

In its simplest form, GPR models the output nonlinearly according to

$$y = f(\mathbf{x}) + \nu, \tag{4}$$

and it follows (1), without assuming that $\mathbf{x}$ and y are linearly related or that $p(y)$ is Gaussian distributed. Nevertheless, it still considers that $p(y|\mathbf{x})$ is Gaussian distributed, i.e., ν is a zero-mean Gaussian. In this way, GP can be understood as a natural nonlinear extension to MMSE estimation. Additionally, GPR does not only estimate (2) from $\mathcal{D}_n$, but it also provides a full statistical description of y given $\mathbf{x}$, particularly

$$p(y|\mathbf{x}, \mathcal{D}_n). \tag{5}$$

GPs can be presented as a nonlinear regressor that expresses the input-output relation in (4) by assuming that a real-valued function $f(\mathbf{x})$, known as latent function, underlies the regression problem and that this function follows a GP. Before the labels are revealed, we assume this latent function has been drawn from a GP prior. GPs are characterized by

their mean and covariance functions, denoted by $\mu(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$, respectively. Even though nonzero mean priors might be of use, working with zero-mean priors typically represents a reasonable assumption and it simplifies the notation. The covariance function explains the correlation between each pair of points in the input space and characterizes the functions that can be described by the GP. For example, $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ only yields linear latent functions and it is used to solve Bayesian linear regression problems, for which the mean of the posterior process coincides with the MMSE solution in (3), as shown in "Connection to MMSE: GPR with a Linear Latent Function." We cover the design of covariance functions in a later section.

GPs PROVIDE THE CORRECT APPROACH TO SOLVE AN MMSE FILTER NONLINEARLY.

For any finite set of inputs $\mathcal{D}_n$, a GP becomes a multidimensional Gaussian defined by its mean (zero in our case) and covariance matrix, $(\mathbf{K}_n)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_n$. The GP prior becomes

$$p(\mathbf{f}_n|\mathbf{X}_n) = \mathcal{N}(0, \mathbf{K}_n), \tag{6}$$

where $\mathbf{f}_n = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)]^\top$ and $\mathbf{X}_n = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$. We want to compute the estimate for a general input $\mathbf{x}$, when the labels for the n training examples, denoted by $\mathbf{y}_n = [y_1, y_2, \ldots, y_n]^\top$, are known. We can analytically compute (5) by using the standard tools of Bayesian statistics: Bayes' rule, marginalization, and conditioning.

We first apply Bayes' rule to obtain the posterior density for the latent function

$$p(f(\mathbf{x}), \mathbf{f}_n|\mathbf{x}, \mathcal{D}_n) = \frac{p(\mathbf{y}_n|\mathbf{f}_n)p(f(\mathbf{x}), \mathbf{f}_n|\mathbf{x}, \mathbf{X}_n)}{p(\mathbf{y}_n|\mathbf{X}_n)}, \tag{7}$$

where $p(f(\mathbf{x}), \mathbf{f}_n|\mathbf{x}, \mathbf{X}_n)$ is the GP prior in (6) extended with a general input $\mathbf{x}$, $p(\mathbf{y}_n|\mathbf{f}_n)$ is the likelihood for the latent function at the training set, in which $\mathbf{y}_n$ is independent of $\mathbf{X}_n$ given the latent function $\mathbf{f}_n$, and $p(\mathbf{y}_n|\mathbf{X}_n)$ is the marginal likelihood or evidence of the model.

The likelihood function is given by a factorized model

$$p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^{n} p(y_i|f(x_i)), \tag{8}$$

because the samples in $\mathcal{D}_n$ are i.i.d.. In turn, for each pair $(f(\mathbf{x}_i), y_i)$, the likelihood is given by (4), therefore

$$p(y_i|f(\mathbf{x}_i)) \sim \mathcal{N}(f(\mathbf{x}_i), \sigma_\nu^2). \tag{9}$$

A Gaussian likelihood function is conjugate to the Gaussian prior, and hence the posterior in (7) is also a multidimensional

CONNECTION TO MMSE: GPR WITH A LINEAR LATENT FUNCTION

If we replace $f(\mathbf{x})$ in (4) with a linear model

$$y = \mathbf{w}^\top \mathbf{x} + \nu,$$

the GP prior over $f(\mathbf{x})$ becomes a spherical-Gaussian prior distribution over $\mathbf{w}$, $p(\mathbf{w}) \sim \mathcal{N}(0, \sigma_\mathbf{w}^2 \mathbf{I})$.

We can now compute the posterior for $\mathbf{w}$, as we did for the latent function in (7)

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} = \frac{p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}),$$

where $p(y_i|\mathbf{x}_i, \mathbf{w})$ is the likelihood. Since the prior and likelihood are Gaussians, so it is the posterior, and its mean and covariance are given by

$$\mu_\mathbf{w} = \frac{1}{\sigma_\nu^2} \Sigma_\mathbf{w} \mathbf{X}^\top \mathbf{y}, \tag{S1a}$$

$$\Sigma_\mathbf{w} = (\mathbf{X}^\top \mathbf{X}/\sigma_\nu^2 + \mathbf{I}/\sigma_\mathbf{w}^2)^{-1}. \tag{S1b}$$

We can readily notice that (S1a) is the sampled version of (3), when the prior variance $\sigma_\mathbf{w}^2$ tends to infinity (i.e., the prior has no effect of the solution). The precision matrix (the inverse covariance) is composed of two terms: the first depends on the data and the other one on the prior over $\mathbf{w}$. The effect of the prior in the mean and covariance fades away, as we have more available data. The estimate for a general input $\mathbf{x}$ is computed as in (10)

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w}, \tag{S2}$$

which is a Gaussian distribution with mean and variance given by

$$\mu_y = \mathbf{x}^\top \mu_\mathbf{w} = \frac{1}{\sigma_\nu^2} \mathbf{x}^\top \Sigma_\mathbf{w} \mathbf{X}^\top \mathbf{y} \tag{S3}$$

$$\sigma_y^2 = \mathbf{x}^\top \Sigma_\mathbf{w} \mathbf{x} + \sigma_\nu^2. \tag{S4}$$

Equations (S3) and (S4) can be, respectively, rewritten as (13a) and (16), if we use the inner product between the $\mathbf{x}_i$ multiplied by the width of the prior over $\mathbf{w}$, i.e., the kernel matrix is given by $\mathbf{K}_n = \mathbf{X}\sigma_\mathbf{w}^2\mathbf{I}\mathbf{X}^\top$. The kernel matrix must include the width of the prior over $\mathbf{w}$, because the kernel matrix represents the prior of the GP and $\sigma_\mathbf{w}^2$ is the prior of the linear Bayesian estimator. By using the Woodbury's identity, it follows that

$$\Sigma_\mathbf{w} = \sigma_\mathbf{w}^2 \mathbf{I} - \sigma_\mathbf{w}^2 \mathbf{I}\mathbf{X}^\top (\sigma_\nu^2 \mathbf{I} + \mathbf{K}_n)^{-1} \mathbf{X} \sigma_\mathbf{w}^2 \mathbf{I}. \tag{S5}$$

Now, by replacing (S5) in (S3) and (S4), we recover (13a) and (16), respectively. These steps connect the estimation of a Bayesian linear model and the nonlinear estimation using a kernel or covariance function without needing to explicitly indicate the nonlinear mapping.

Gaussian, which simplifies the computations to obtain (5). If the observation model were not Gaussian, warped GPs could be used to estimate (5).

Finally, we can obtain the posterior density in (5) for a general input $\mathbf{x}$ by conditioning on the training set and $\mathbf{x}$, and by marginalizing the latent function

$$p(y|\mathbf{x}, \mathcal{D}_n) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathbf{x}, \mathcal{D}_n)\,df(\mathbf{x}), \tag{10}$$

where

$$p(f(\mathbf{x})|\mathbf{x}, \mathcal{D}_n) = \int p(f(\mathbf{x}), \mathbf{f}_n|\mathbf{x}, \mathcal{D}_n)\,d\mathbf{f}_n. \tag{11}$$

We have divided the marginalization in two separate equations to show the marginalization of the latent function over the training set in (11), and the marginalization of the latent function at a general input $\mathbf{x}$ in (10). As mentioned earlier, the likelihood and the prior are Gaussians and therefore the marginalization in (10) and (11) only involves Gaussian distributions. Thereby, we can analytically compute (10) and (11) by using Gaussian conditioning and marginalization properties, leading to the following Gaussian density for the output:

$$p(f(\mathbf{x})|\mathbf{x}, \mathcal{D}_n) \sim \mathcal{N}(\mu_{f(\mathbf{x})}, \sigma^2_{f(\mathbf{x})}), \tag{12}$$

where

$$\mu_{f(\mathbf{x})} = \mathbf{k}^\top \mathbf{C}_n^{-1}\mathbf{y}_n, \tag{13a}$$
$$\sigma^2_{f(\mathbf{x})} = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^\top \mathbf{C}_n^{-1}\mathbf{k}, \tag{13b}$$

with

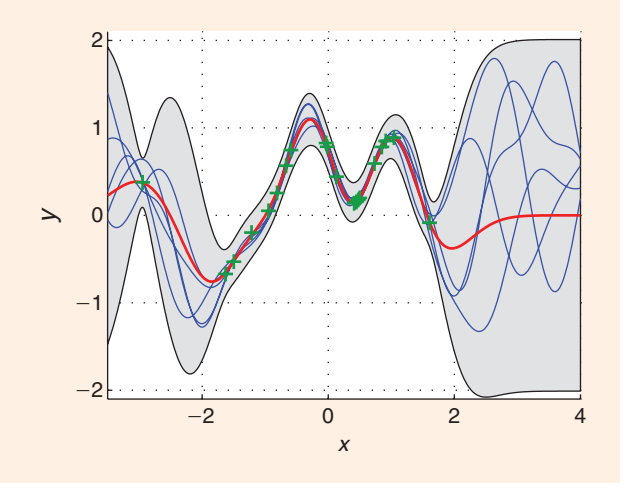
$$\mathbf{k} = [k(\mathbf{x}_1, \mathbf{x}), k(\mathbf{x}_2, \mathbf{x}), \ldots, k(\mathbf{x}_n, \mathbf{x})]^\top, \tag{14}$$
$$\mathbf{C}_n = \mathbf{K}_n + \sigma_\nu^2 \mathbf{I}_n. \tag{15}$$

The mean for $p(y|\mathbf{x}, \mathcal{D}_n)$ is also given by (13a), i.e., $\mu_y = \mu_{f(\mathbf{x})}$, and its variance is

$$\sigma_y^2 = \sigma^2_{f(\mathbf{x})} + \sigma_\nu^2, \tag{16}$$

which, as expected, also accounts for the noise in the observation model.

The mean prediction of GPR in (13a) is the solution provided by KLS, or kernel ridge regression (KRR) [7], in which the covariance function takes the place of the kernel. However, unlike standard kernel methods, GPR provides error bars for each estimate in (13b) or (16) and has a natural procedure for setting the covariance/kernel by evidence sampling or maximization, as detailed in the section "Covariance Functions." In SVM or KRR, the hyperparameters are typically adjusted by cross-validation, needing to retrain the models for different settings of the hyperparameters on a grid search. So, typically only one or two hyperparameters can be fitted. GPs can actually learn tens of hyperparameters, because either sampling or evidence maximization allows setting them by a hassle-free procedure.



[FIG2] An example of a GP posterior in (12) with 20 training samples, denoted by green +. Five instances of the posterior are plotted by thin blue lines and the mean of the posterior, μ_y, by a red thick line. The shaded area denotes the error bars for the mean prediction: $\mu_y \pm 2\sigma_y$.

AN EXAMPLE

In Figure 2, we include an illustrative example with 20 training points, in which we depict (12) for any $\mathbf{x}$ between -3 and 4. We used standard functions from the GPML toolbox, available at http://www.gaussianprocess.org/gpml/, to generate the GP in this figure. We have chosen a Gaussian kernel that is fixed as $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-2\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ and $\sigma_\nu = 0.1$. In the plot, we show the mean of the process in red and the shaded area denotes the error bar for each prediction, i.e., $\mu_y \pm 2\sigma_y$. We also plot five samples from the posterior in thin blue lines.

We observe three different regions in the figure. On the right-hand side, we do not have samples and, for $\mathbf{x} > 3$, the GPR provides the solution given by the prior (zero mean and ± 2). At the center, where most of the data points lie, we have a very accurate view of the latent function with small error bars (close to $\pm 2\sigma_\nu$). On the left-hand side, we only have two samples, and we notice the mixed effect of the prior widening the error bars and the data points constraining the values of the mean to lie close to the available samples. This is the typical behavior of GPR, which provides an accurate solution where the data lies and high error bars where we do not have available information and, consequently, we presume that the prediction in that area is not accurate.

RECURSIVE GPs

In many signal processing applications, the samples become available sequentially and estimation algorithms should obtain the new solution every time a new datum is received. To keep the computational complexity low, it is more interesting to perform inexpensive recursive updates rather than to recalculate

the entire batch solution. Online GPs [8] fulfill these requisites as follows.

Let us assume that we have observed the first n samples and that at this point the new datum $\mathbf{x}_{n+1}$ is provided. We can readily compute the predicted distribution for y_{n+1} using (13a), (13b), and (16). Furthermore, by using the formula for the inverse of a partitioned matrix and the Woodbury identity, we update $\mathbf{C}_{n+1}^{-1}$ from $\mathbf{C}_n^{-1}$

$$\mathbf{C}_{n+1}^{-1} = \begin{bmatrix} \mathbf{C}_n^{-1} + \mathbf{C}_n^{-1}\mathbf{k}_{n+1}\mathbf{k}_{n+1}^\top \mathbf{C}_n^{-1}/\sigma_{y_{n+1}}^2 & -\mathbf{C}_n^{-1}\mathbf{k}_{n+1}/\sigma_{y_{n+1}}^2 \\ -\mathbf{k}_{n+1}^\top \mathbf{C}_n^{-1}/\sigma_{y_{n+1}}^2 & 1/\sigma_{y_{n+1}}^2 \end{bmatrix}, \quad (17)$$

where $\sigma_{y_{n+1}}^2$ and $\mathbf{k}_{n+1}$ correspond to (16) and (14), respectively, for $\mathbf{x} = \mathbf{x}_{n+1}$.

Nevertheless, for online scenarios, it is more convenient to update the predicted mean and covariance matrix for all the available samples, as it is easier to interpret how the prediction changes with each new datum. Additionally, as we will show in the section "Tracking Nonstationary Scenarios: Learning to Forget," this formulation makes the adaptation to nonstationary scenarios straightforward. Let us denote by $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$ the posterior mean and covariance matrix for the samples in $\mathcal{D}_n$. By applying (13a) and (13b) we obtain

$$\boldsymbol{\mu}_n = \mathbf{K}_n \mathbf{C}_n^{-1} \mathbf{y}_n, \quad (18a)$$

$$\boldsymbol{\Sigma}_n = \mathbf{K}_n - \mathbf{K}_n \mathbf{C}_n^{-1} \mathbf{K}_n. \quad (18b)$$

Once the new datum $(\mathbf{x}_{n+1}, y_{n+1})$ is observed, the updated mean and covariance matrix can be computed recursively as follows:

$$\boldsymbol{\mu}_{n+1} = \begin{bmatrix} \boldsymbol{\mu}_n \\ \mu_{f(\mathbf{x}_{n+1})} \end{bmatrix} - \frac{\mu_{f(\mathbf{x}_{n+1})} - y_{n+1}}{\sigma_{y_{n+1}}^2} \begin{bmatrix} \mathbf{h}_{n+1} \\ \sigma_{f(\mathbf{x}_{n+1})}^2 \end{bmatrix} \quad (19a)$$

$$\boldsymbol{\Sigma}_{n+1} = \begin{bmatrix} \boldsymbol{\Sigma}_n & \mathbf{h}_{n+1} \\ \mathbf{h}_{n+1}^\top & \sigma_{f(\mathbf{x}_{n+1})}^2 \end{bmatrix} - \frac{1}{\sigma_{y_{n+1}}^2} \begin{bmatrix} \mathbf{h}_{n+1} \\ \sigma_{f(\mathbf{x}_{n+1})}^2 \end{bmatrix} \begin{bmatrix} \mathbf{h}_{n+1}^\top & \sigma_{f(\mathbf{x}_{n+1})}^2 \end{bmatrix}, \quad (19b)$$

where $\mathbf{h}_{n+1} = \boldsymbol{\Sigma}_n \mathbf{K}_n^{-1} \mathbf{k}_{n+1} = (\mathbf{I}_n - \mathbf{K}_n \mathbf{C}_n^{-1}) \mathbf{k}_{n+1}$. As can be observed in (19a), the mean of the new process is obtained by applying a correction term to the previous mean, proportional to the estimation error, $\mu_{f(\mathbf{x}_{n+1})} - y_{n+1}$. Only either $\boldsymbol{\Sigma}_n$ or $\mathbf{C}_n^{-1}$ needs to be stored and updated in an online formulation because of the relation between them stated in (18b).

The recursive update of the mean in (19a) is equivalent to what is known as *kernel recursive least-squares* (KRLS) in the signal processing literature (see, for instance, [8]–[10]). The unbounded growth of the involved matrices, visible in (19) and (17), is the main limitation in the KRLS formulation. Practical KRLS implementations typically either limit this growth [10], [11] or even fix the matrix sizes [12]. Nevertheless, the solution of KRLS is limited to the mean only and it cannot estimate confidence intervals. By using a GP framework, though, an estimate of the entire posterior distribution is obtained, including the covariance in (19b).

COVARIANCE FUNCTIONS

In the previous sections, we have assumed that the covariance functions $k(\mathbf{x}, \mathbf{x}')$ are known, which is not typically the case. In fact, the design of a good covariance function is crucial for GPs to provide accurate nonlinear solutions. The covariance function plays the same role as the kernel function in SVMs or KLS [7]. It describes the relation between the inputs and its form determines the possible solutions of the GPR. It controls how fast the function can change or how the samples in one part of the input space affect the latent function everywhere else. For most problems, we can specify a parametric kernel function that captures any available information about the problem at hand. As previously discussed, unlike kernel methods, GPs can infer these parameters, the so-called *hyperparameters*, from the samples in $\mathcal{D}_n$ using the Bayesian framework. Instead of relying on computational intensive procedures as cross-validation [13] or learning the kernel matrix [14], as kernel methods need to.

The covariance function must be positive semidefinite, as it represents the covariance matrix of a multidimensional Gaussian distribution. The covariance can be built by adding simpler covariance matrices, weighted by a positive hyperparameter, or by multiplying them together, as the addition and multiplication of positive definite matrices yields a positive definite matrix. In general, the design of the kernel should rely on the information that we have for each estimation problem and should be designed to get the most accurate solution with the least amount of samples. Nevertheless, the following kernel in (20) often works well in signal processing applications

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha_1 \exp\left(-\sum_{\ell=1}^{d} \gamma_\ell \| x_{i\ell} - x_{j\ell} \|^2 \right) + \alpha_2 \mathbf{x}_i^\top \mathbf{x}_j + \alpha_3 \delta_{ij}, \quad (20)$$

where $\boldsymbol{\theta} = [\alpha_1, \gamma_1, \gamma_2, \ldots, \gamma_d, \alpha_2, \alpha_3]^\top$ are the hyperparameters. The first term is a radial basis kernel, also denoted as RBF or Gaussian, with a different length-scale for each input dimension. This term is universal and allows constructing a generic nonlinear regressor. If we have symmetries in our problem, we can use the same length-scale for all dimensions: $\gamma_\ell = \gamma$ for $\ell = 1, \ldots, d$. The second term is the linear covariance function. The last term represents the noise variance $\alpha_3 = \sigma_v^2$, which can be treated as an additional hyperparameter to be learned from the data. We can add other terms or other covariance functions that allow for faster transitions, like the Matérn kernel among others [5].

If the hyperparameters, $\boldsymbol{\theta}$, are unknown, the likelihood in (8) and the prior in (6) can, respectively, be expressed as $p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta})$ and $p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$, and we can proceed to integrate out $\boldsymbol{\theta}$ as we did for the latent function, $\mathbf{f}$, in the section "Gaussian Processes for Regression." We have dropped the subindex n, as it is inconsequential and unnecessarily clutters the notation. First, we compute the marginal likelihood of the hyperparameters of the kernel given the training data set

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) \, d\mathbf{f}. \quad (21)$$

Second, we can define a prior for the hyperparameters, $p(\boldsymbol{\theta})$, that can be used to construct its posterior. Third, we integrate out the hyperparameters to obtain the predictions. However, in this case, the marginal likelihood does not have a conjugate prior and the posterior cannot be obtained in closed form. Hence, the integration has to be done either by sampling or approximations. Although this approach is well principled, it is computational intensive and it may be not feasible for some applications. For example, Markov-chain Monte Carlo (MCMC) methods require several hundred to several thousand samples from the posterior of $\boldsymbol{\theta}$ to integrate it out. Interested readers can find further details in [5].

Alternatively, we can maximize the marginal likelihood in (21) to obtain its optimal setting [1]. Although setting the hyperparameters by maximum likelihood (ML) is not a purely Bayesian solution, it is fairly standard in the community and it allows using Bayesian solutions in time-sensitive applications. This optimization is nonconvex [15], but, as we increase the number of training samples, the likelihood becomes a unimodal distribution around the ML hyperparameters and the solution can be found using gradient ascent techniques. See [5] for further details.

MANY LOW-COST APPROXIMATIONS CAN BE EXPRESSED AS EXACT INFERENCE UNDER A MODIFIED PRIOR.

SPARSE GPs: DEALING WITH LARGE-SCALE DATA SETS

To perform inference under any GP model, the inverse of the covariance matrix must be computed. This is a costly operation, $\mathcal{O}(n^3)$, that becomes prohibitive for large enough n. Given the ever-increasing availability of large-scale databases, a lot of effort has been devoted over the last decade to the development of approximate methods that allow inference in GPs to scale linearly with the number of data points. These approximate methods are referred to as "sparse GPs," since they approximate the full GP model using a finite-basis-set expansion. This set of bases is usually spawned by using a common functional form with different parametrizations. For instance, it is common to use bases of the type $\{k(\mathbf{z}_b, \mathbf{x})\}_{b=1}^m$, where $\{\mathbf{z}_b\}_{b=1}^m$, known as the *active set*, is a subset of the input samples parametrizing the bases.

Under the unifying framework of [16], it can be shown that most relevant sparse GP proposals [17], [18], which were initially thought of as entirely different low-cost approximations, can be expressed as exact inference under different modifications of the original GP prior. This modified prior induces a rank-m ($m \ll n$) covariance matrix—plus optional (block) diagonal correcting terms—clarifying how the reduced $\mathcal{O}(m^2 n)$ cost of exact inference arises.

Among the mentioned approximations, the sparse pseudoinput GP (SPGP) [18] is generally regarded as the most efficient. Unlike other alternatives, it does not require the active set to be a subset of the training data. Instead, $\{\mathbf{z}_b\}_{b=1}^m$ can be selected to lie anywhere in the input space, thus increasing the flexibility of the finite set expansion. This selection is typically performed by evidence maximization. An even more flexible option, which does not require the active set to even lie in the input domain, is presented in [19].

Despite the success of SPGP, it is worth mentioning that increasing the number of bases in this algorithm does not yield, in general, convergence to the full GP solution because the active set $\{\mathbf{z}_b\}_{b=1}^m$ is not constrained to be a subset of input data. This might lead to overfitting in some pathological cases. A recent variational sparse GP proposal that guarantees convergence to the full GP solution while still allowing the active set to be unconstrained is presented in [20].

Further approaches yielding reduced computational cost involve numerical approximations to accelerate matrix-vector multiplications and compactly supported covariance functions that set most entries of the covariance matrix to zero [21].

Sparsity is often seen in online signal processing in the form of pruning, which restricts the active set to a subset of input data. The success of SPGP and its variational counterpart suggests that advanced forms of pruning may result in increased efficiency for a given sparsity level.

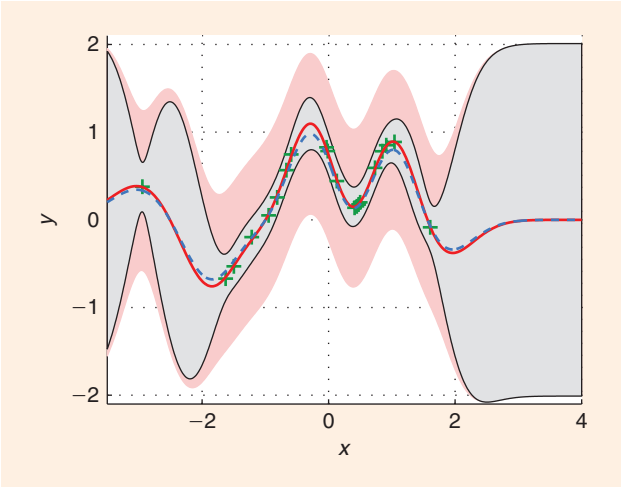
WARPED GPs: BEYOND THE STANDARD NOISE MODEL

Even though GPs are very flexible priors for the latent function, they might not be suitable to model all types of data. It is often the case that applying a logarithmic transformation to the target variable of some regression task (e.g., those involving stock prices and measured sound power) can enhance the ability of GPs to model it.

In [22], it is shown that it is possible to include a nonlinear preprocessing of output data $h(y)$ (called warping function in this context) as part of the modeling process and learn it. In more detail, a parametric form for $z = h(y)$ is selected, then z (which depends on the parameters of $h(y)$) is regarded as a GP, and finally, the parameters of $h(y)$ are selected by maximizing the evidence of such GP (i.e., an ML approach). The authors suggest using $h(y) = \sum_{i=1}^{l} a_i \tanh(b_i(y + c_i))$ as the parametric form of the warping function, but any option resulting in a monotonic function is valid. A nonparametric version of warped GPs using a variational approximation has been proposed in [23].

TRACKING NONSTATIONARY SCENARIOS: LEARNING TO FORGET

KRLS algorithms, discussed in the section "Recursive GPs," traditionally consider that the mapping function $f(\cdot)$ is constant throughout the whole learning process [10], [24]. However, in the signal processing domain, this function (which might represent, for instance, a fading channel) is often subject to changes and the model must account for this nonstationarity. Some kernel-based algorithms have been proposed to deal with nonstationary scenarios. They include a kernelized version of the extended RLS filter [24], a sliding-window KRLS approach [12] and a family of projection-based algorithms [25], [26].



[FIG3] An illustration of forgetting step (22) on the GP of Figure 2. The dashed line represents the predictive mean that is pulled toward the prior mean, while the shaded red area represents the region $\mu_y \pm 2\sigma_y$ after forgetting.

To add adaptivity to the online GP algorithm described in the section "Recursive GPs," it is necessary to make it "forget" the information contained in old samples. This becomes possible by including a "forgetting" step after each update

$$\boldsymbol{\mu} \leftarrow \sqrt{\lambda}\,\boldsymbol{\mu} \tag{22a}$$

$$\boldsymbol{\Sigma} \leftarrow \lambda\boldsymbol{\Sigma} + (1-\lambda)\mathbf{K} \tag{22b}$$

to shift the posterior distribution toward the prior (for (for $0 < \lambda < 1$), thus effectively reducing the influence of older samples. Note that when using this formulation there is no need to store or update $\mathbf{C}^{-1}$; see [9] for further details. The adaptive, GP-based algorithm obtained in this manner is known as KRLS-T.

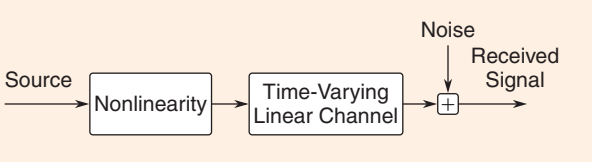
Equation (22) might seem like an ad hoc step to enable forgetting. However, it can be shown that the whole learning procedure—including the mentioned controlled forgetting step—corresponds exactly to a principled nonstationary scheme within the GP framework, as described in [27]. It is sufficient to consider an augmented input space that includes the time stamp t of each sample and define a spatiotemporal covariance function

$$k_{\mathrm{st}}([t\ \mathbf{x}^\top]^\top, [t'\ \mathbf{x}'^\top]^\top) = k_t(t,t')k_s(\mathbf{x},\mathbf{x}'), \tag{23}$$

where $k_s(\mathbf{x},\mathbf{x}')$ is the already-known spatial covariance function and $k_t(t,t')$ is a temporal covariance function giving more weight to samples that are closer in time. Inference on this augmented model effectively accounts for nonstationarity in $f(\cdot)$, and recent samples have more impact in predictions for the current time instant. It is fairly simple to include this augmented model in the online learning process described in the section "Recursive GPs." When the temporal covariance is set to $k_t(t,t') = \lambda^{(|t-t'|)/2}$, $\lambda \in (0,1]$, inference in the augmented spatiotemporal GP model is exactly equivalent to using (22) after each update (19) in the algorithm of the section "Recursive GPs," which has the added benefit of being inexpensive and online. See [9], [27], and [28] for further details.

Observe that λ is used here to model the speed at which $f(\cdot)$ varies, playing a similar role to that of the forgetting factor in linear adaptive filtering algorithms. When used with a linear spatial covariance, the above model reduces to linear extended RLS filtering. The selection of this parameter is usually rather ad hoc. However, using the GP framework, we can select it in a principled manner using Type-II ML; see [27].

In Figure 3, we take the example of Figure 2 and we apply a forgetting factor $\lambda = 0.8$. The red continuous line indicates the original mean function before forgetting. After applying one forgetting update, this mean function is displaced toward zero, as indicated by the blue dashed line. The shaded gray area represents the error bars prior to forgetting. The forgetting update expands this area into the shaded red area, which tends to the prior variance of one.



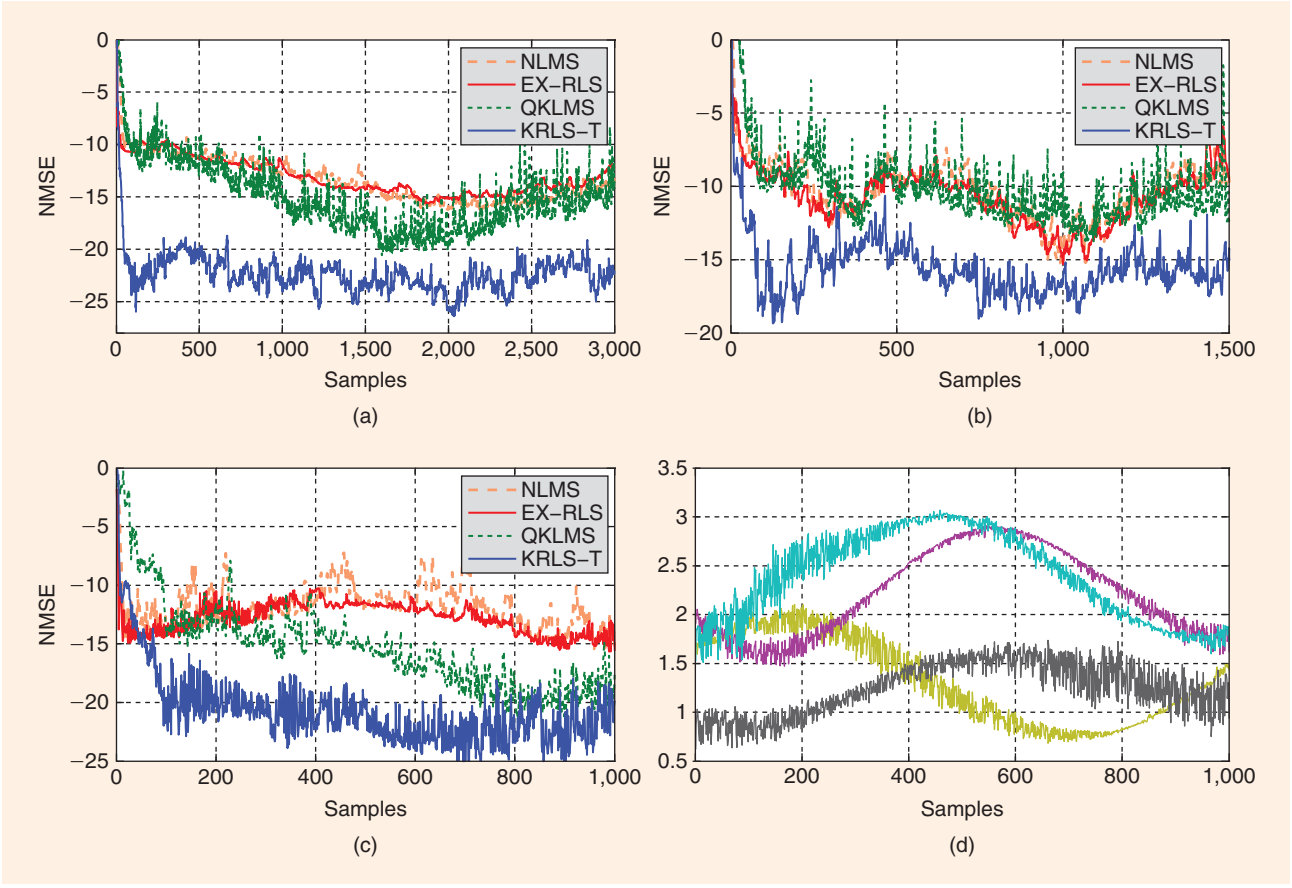
[FIG4] The nonlinear channel used in the example consists of a nonlinearity followed by a linear channel.

TRACKING A TIME-SELECTIVE NONLINEAR COMMUNICATION CHANNEL

To illustrate the validity of the adaptive filtering algorithm, we focus on the problem of tracking a nonlinear Rayleigh fading channel [29, Ch. 7]. The used model consists of a memoryless saturating nonlinearity followed by a time-varying linear channel, as shown in Figure 4. This model appears, for instance, in broadcast or satellite communications when the amplifier operates close to saturation regime [30].

In a first, simulated setup, the time-varying linear fading channel consists of five randomly generated paths, and the saturating nonlinearity is chosen as $y = \tanh(x)$. We fix the symbol rate at $T = 1\mu$s, and we simulate two scenarios: one with a normalized Doppler frequency of $f_d T = 10^{-4}$ (where f_d denotes the Doppler spread), representing a slow-fading channel, and another one with $f_d T = 10^{-3}$, corresponding to a fast time-varying channel. Note that a higher Doppler frequency yields a more difficult tracking problem, as it corresponds to a channel that changes faster in time. We consider a Gaussian source signal, and we add 30 dB of additive white Gaussian noise to the output signal. Given one input-output data pair per time instant, the tracking problem consists in estimating the received signal that corresponds to a new channel input.

Figure 5(a) and (b) illustrates the tracking results obtained by KRLS-T in these scenarios. As a reference, we include the performance of several state-of-the-art adaptive filtering algorithms whose MATLAB implementations are taken from the Kernel Adaptive Filtering Toolbox, which is available at http://sourceforge.net/projects/kafbox/. In particular, we compare



[FIG5] Tracking results on a nonlinear Rayleigh fading channel. (a) Simulation results for a slow-fading scenario. (b) Simulation results for a fast time-varying scenario. (c) Tracking results on data measured on the test bed with fast time-varying channels. (d) Channel taps of the noisy linear channels, measured on the test bed setup.

KRLS-T with normalized least mean squares (NLMS), extended RLS (EX-RLS), both of which are linear algorithms (see [29]), and quantized kernel LMS (QKLMS) [31], which is an efficient, kernelized version of the LMS algorithm. A Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ is used for QKLMS and KRLS-T. In each scenario, the optimal hyperparameters of KRLS-T are obtained by performing Type-II ML optimization (see the section "Covariance Functions") on a separate data set of 500 test samples. The optimal parameters of the other algorithms are obtained by performing cross-validation on the test data set. To avoid an unbounded growth of the matrices involved in KRLS-T, its memory is limited to 100 bases that are selected by pruning the least relevant bases (see [9] for details on the pruning mechanism). The quantization parameter of QKLMS is set to yield similar memory sizes. As can be seen in Figure 5(a) and (b), KRLS-T outperforms the other algorithms with a significant margin in both scenarios. By being kernel based, it is capable to deal with nonlinear identification problems, in contrast to the classical EX-RLS and NLMS algorithms. Furthermore, it shows excellent convergence speed and steady-state performance when compared to QKLMS. Additional experimental comparisons to other kernel adaptive filters can be found in [9].

In a second setup, we used a wireless communication test bed that allows the evaluation of the performance of digital communication systems in realistic indoor environments. This platform is composed of several transmit and receive nodes, each one including a radio-frequency front end and baseband hardware for signal generation and acquisition. The front end also incorporates a programmable variable attenuator to control the transmit power value and therefore the signal saturation. A more detailed description of the test bed can be found in [32]. Using the hardware platform, we reproduced the model corresponding to Figure 4 by transmitting clipped orthogonal frequency-division multiplexing (OFDM) signals centered at 5.4 GHz over real frequency-selective and time-varying channels. Notice that, unlike the simulated setup, several parameters such as the noise level and the variation of the channel coefficients are unknown. To have an idea about the channel characteristics, we first measured the indoor channel using the procedure described in [32]. As an example, the variation of the four main channel coefficients is depicted in Figure 5(d), indicating a normalized Doppler frequency around $f_d T = 10^{-3}$. We then transmitted periodically OFDM signals with the transmit amplifier operating close to saturation and acquired the

[TABLE 1] STEADY-STATE NMSE PERFORMANCE FOR FIGURE 5.

	NLMS	EX-RLS	QKLMS	KRLS-T
$f_dT = 10^{-4}$, SIMULATED	−13.3 dB	−13.0 dB	−14.6 dB	−22.3 dB
$f_dT = 10^{-3}$, SIMULATED	−10.6 dB	−11.0 dB	−9.9 dB	−15.3 dB
$f_dT = 10^{-3}$, REAL DATA	−11.5 dB	−12.5 dB	−15.8 dB	−21.3 dB

received signals. The transmitted and received signals were used to track the nonlinear channel variations as in the simulated setup. The results, shown in Figure 5(c), are similar to those of the simulated setup. Finally, the steady-state NMSE performances of all three scenarios, Figure 5(a)–(c), are summarized in Table 1.

GPs FOR CLASSIFICATION

For classification problems, the labels are drawn from a finite set and GPs return a probabilistic prediction for each label in the finite set, i.e., how certain is the classifier about its prediction. In this tutorial, we limit our presentation of GPs for classification (GPC) for binary classification problems, i.e., $y_i \in \{0,1\}$. For GPC, we change the likelihood model for the latent function at x using a response function $\Phi(\cdot)$

$$p(y = 1 \,|\, f(\mathrm{x})) = \Phi(f(\mathrm{x})). \tag{24}$$

The response function "squashes" the real-valued latent function to an $(0,1)$-interval that represents the posterior probability for y [5]. Standard choices for the response function are $\Phi(a) = 1/(1+\exp(-a))$ and the cumulative density function of a standard normal distribution, used in logistic and probit regression, respectively.

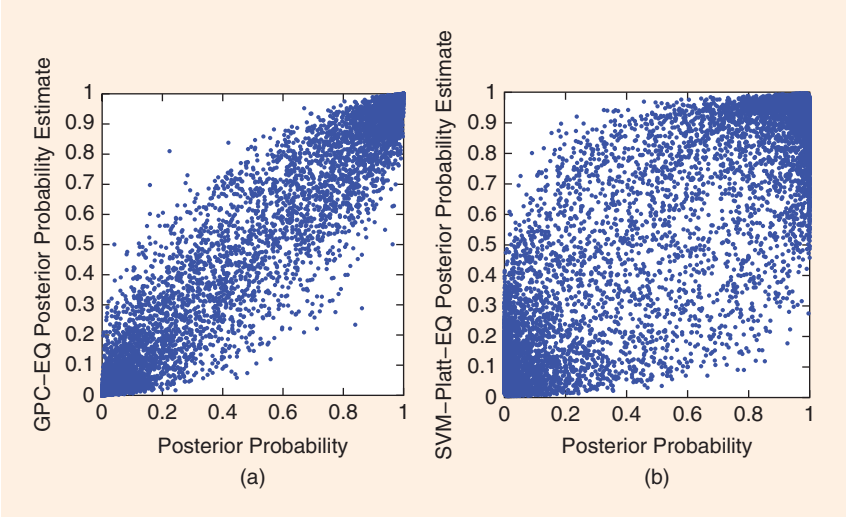
GPs CAN BE SOLVED ITERATIVELY WITH AN RLS FORMULATION THAT CAN BE ADAPTED TO NONSTATIONARY ENVIRONMENTS EFFICIENTLY.

The integrals in (10) and (11) are now analytically intractable, because the likelihood and the prior are not conjugated. Therefore, we have to resort to numerical methods or approximations to solve them. The posterior distribution in (7) is typically single-mode and the standard methods approximate it with a Gaussian [5]. Using a Gaussian approximation for (7) allows exact marginalization in (11), and we can use numerical integration for solving (10), as it involves marginalizing a single real-valued quantity. The two standard approximations are the Laplace method or expectation propagation (EP) [33]. In [2], EP is shown to be a more accurate approximation.

PROBABILISTIC CHANNEL EQUALIZATION

GPC predictive performance is similar to other nonlinear discriminative methods, such as SVMs. However, if the probabilistic output is of importance, then GPC outperforms other kernel algorithms, because it naturally incorporates the confidence interval in its predictions. In digital communication, channel decoders follow equalizers, which work optimally when accurate posterior estimates are given for each symbol. To illustrate that GPC provide accurate posterior probability estimates, we equalize a dispersive channel model like the one in Figure 4, using GPC and SVM with a probabilistic output. These outputs are subsequently fed to a low-density parity-check (LDPC) belief-propagation-based channel decoder to assess the quality of the estimated posterior probabilities. Details for the experimental setup can be found in [34] in which linear and nonlinear channel models are tested. We now summarize the results for the linear channel model in that paper.

In Figure 6, we depict the posterior probability estimates versus the true posterior probability, in (a) for the GPC-based equalizer and in (b) for SVM-based equalizer, to emphasize the differences between the equalizers we use a highly noisy scenario with normalized signal-to-noise ratio of 2 dB. If we threshold at 0.5, both equalizers provide similar error rates and



[FIG6] GPC as probabilistic channel equalizer. (a) Calibration curve for the GPC and (b) calibration curve for the SVM.

we cannot tell if there is an advantage from using GPC. However, if we consider the whole probability space, GPC predictions are significantly closer to the main diagonal that represents a perfect match, hence GPC provides more accurate predictions to the channel decoder.

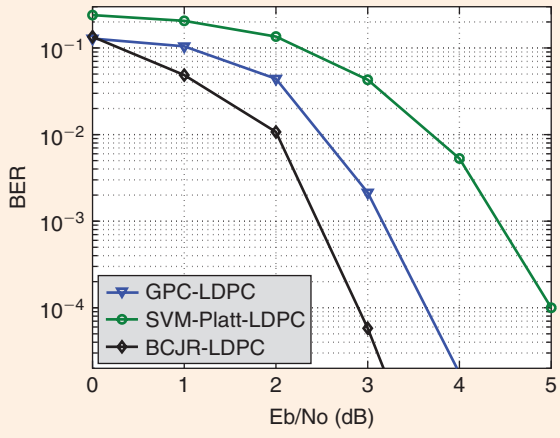
To further quantify the gain from using a GPC-based equalizer with accurate posterior probability estimates, we plot the bit error rate (BER) in Figure 7 after the probabilistic channel encoder, in which the GPC-based equalizer clearly outperforms the SVM-based equalizer and is close to the optimal solution [known channel and forward-backward (BCJR) equalizer]. This example is illustrative of the results that can be expected from GPC when a probabilistic output is needed to perform optimally.

DISCUSSION

In this tutorial, we have presented GPR in detail from the point of view of MMSE/Wiener filtering, so it is amenable to signal processing practitioners. GPR provides the same mean estimate as KLS or KRR for the same kernel matrix. On the plus side, GPR provides error bars that take into account the approximation error and the error from the likelihood model, so we know the uncertainty of our model for any input point (see Figure 2), while KLS assumes the error bars are given by the likelihood function (i.e., constant for the whole input space). Additionally, GPR naturally allows computing the hyperparameters of the kernel function by sampling or maximizing the marginal likelihood, being able to set tens of hyperparameters, while KLS or SVM need to rely on cross-validation, in which only one or two parameters can be easily tuned. On the minus side, the GP prior imposes a strong assumption on the error bars that might not be accurate, if the latent variable model does not follow a GP. Although, in any case, it is better than not having error bars.

We have also shown that some of the limitations of the standard GPR can be eased. GPs can be extended to non-Gaussian noise models and classification problems, in which GPC provides an accurate a posteriori probability estimate. The computational complexity of GPs can be reduced considerably, from cubic to linear in the number of training examples, without significantly affecting the mean and error bars prediction. Finally, GPs can be solved iteratively, with an RLS formulation that can be adapted to nonstationary environments efficiently.

Instead of covering more methods and applications in detail, our intention was to provide a tutorial article on how to use GPs in signal processing, with a number of illustrative examples. Nevertheless, since we assume that there are several other methods and applications that are relevant to the reader, we finish with a brief list of further topics. In particular, GPs have also been applied to problems including modeling human motion [35], source separation [36], estimating chlorophyll concentration [37], approximating stochastic differential equations [38] and multiuser detection [39], among others.



[FIG7] GPC and SVM as probabilistic channel equalizer in channel LDPC decoding: BER for the GPC-LDPC (∇), the SVM-LDPC (○), and the optimal solution (◇).

ACKNOWLEDGMENTS

The authors would like to thank Jesús Gutiérrez, of the University of Cantabria, Spain, for his assistance in capturing the data of the test bed experiment. This work has been partially supported by TEC2012-38800-C03-{01,02} (ALCIT), TEC2010-19545-C04-03 (COSIMA), TEC2009-14504-C02-{01,02} (DEIPRO), Consolider-Ingenio 2010 CSD2008-00010 (COMONSENS), and MLPM2012 (UE- FP7-PEOPLE-ITN).

AUTHORS

Fernando Pérez-Cruz (fernando@tsc.uc3m.es) received a Ph.D. degree in electrical engineering in 2000 from Polytechnic University in Madrid. He is an associate professor with the Department of Signal Theory and Communication at the University Carlos III in Madrid. He has been a visiting professor at Princeton University (Marie Curie Fellow) and has held positions at the Gatsby Unit of University College London, the Max Planck Institute for Biological Cybernetics, and BioWulf Technologies. His current research interest lies in machine learning and information theory and its application to signal processing and communications. He is a Senior Member of the IEEE.

Steven Van Vaerenbergh (steven@gtas.dicom.unican.es) received the M.Sc. degree in electrical engineering from Ghent University, Belgium, in 2003, and the Ph.D. degree from the University of Cantabria, Santander, Spain, in 2010. He was a visiting researcher with the Computational Neuroengineering Laboratory, University of Florida, Gainesville, in 2008. Currently, he is a postdoctoral associate with the Department of Telecommunication Engineering, University of Cantabria, Spain. His main research interests include machine learning and its applications to adaptive filtering, target tracking, and system identification. He is a Member of the IEEE.

Juan José Murillo-Fuentes (murillo@us.es) received his telecommunication engineering degree in 1996 from the Universidad

de Sevilla and his Ph.D. degree in telecommunication engineering in 2001 from the Universidad Carlos III de Madrid, Spain. He is currently an associate professor at the Universidad de Sevilla, teaching wireless communications and signal processing. He is also a professor of the Vodafone Master. His research interests lie in algorithm development for signal processing and machine learning and their application to watermarking, predistortion, jamming detection, and channel decoding. He is a Senior Member of the IEEE.

Miguel Lázaro-Gredilla (miguel@tsc.uc3m.es) received the telecommunication engineering degree (Honors) from the University of Cantabria, Santander, Spain, and the Ph.D. degree (Honors) from the Universidad Carlos III de Madrid, Leganés, Spain, in 2004 and 2010, respectively. He has been a visiting researcher at the University of Cambridge, United Kingdom; the University of Manchester, United Kingdom; and the University of Cantabria. Currently, he is an assistant professor at Universidad Carlos III de Madrid. His current research interests include GPs, Bayesian models, and approximate inference. He is a Member of the IEEE.

Ignacio Santamaría (nacho@gtas.dicom.unican.es) received the telecommunication engineering degree and the Ph.D. degree in electrical engineering from the Polytechnic University of Madrid, Spain, in 1991 and 1995, respectively. He joined the Department of Telecommunications Engineering, University of Cantabria, in 1992, where he is currently a full professor. He has coauthored more than 150 publications in refereed journals and international conference papers. He is currently an associate editor of *IEEE Transactions on Signal Processing*. His current research interests include signal-processing algorithms for wireless communication systems, multivariate statistical techniques, and machine-learning theories. He is a Senior Member of the IEEE.

REFERENCES

[1] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Neural Information Processing Systems 8*. Cambridge, MA: MIT Press, 1996, pp. 598–604.

[2] M. Kuss and C. Rasmussen, "Assessing approximate inference for binary Gaussian process classification," *Mach. Learn. Res.*, vol. 6, pp. 1679–1704, Oct. 2005.

[3] N. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *Mach. Learn. Res.*, vol. 6, pp. 1783–1816, Nov. 2005.

[4] A. O'Hagan and J. F. Kingman, "Curve fitting and optimal design for prediction," *J. Roy. Stat. Soc. Series B,* vol. 40, no. 1, pp. 1783–1816, 1978.

[5] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[6] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[7] F. Pérez-Cruz and O. Bousquet, "Kernel methods and their potential use in signal processing," *IEEE Signal Processing Mag.*, vol. 21, no. 3, pp. 57–65, 2004.

[8] L. Csató and M. Opper, "Sparse representation for Gaussian process models," in *Neural Information Processing Systems 13*. Cambridge, MA: MIT Press, 2001, pp. 444–450.

[9] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Trans. Neural Networks Learn. Syst.,* vol. 23, no. 8, pp. 1313–1326, Aug. 2012.

[10] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares algorithm," *IEEE Trans. Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[11] W. Liu, I. Park, and J. C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Networks*, vol. 20, no. 12, pp. 1950–1961, 2009.

[12] S. Van Vaerenbergh, J. Vía, and I. Santamaría, "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP),* Toulouse, France, May 2006, vol. 5, pp. 789–792.

[13] G. S. Kimeldorf and G. Wahba, "Some results on Tchebycheffian spline functions," *J. Math. Anal. Appl.,* vol. 33, no. 1, pp. 82–95, 1971.

[14] O. Bousquet and D. J. L. Herrmann, "On the complexity of learning the kernel matrix," in *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press, 2003, pp. 399–406.

[15] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[16] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Mach. Learn. Res.*, vol. 6, pp. 1939–1959, Dec. 2005.

[17] M. Seeger, C. K. I. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *Proc. 9th Int. Workshop Artificial Intelligence and Statistics*, 2003.

[18] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 1259–1266.

[19] M. Lázaro-Gredilla and A. Figueiras-Vidal, "Inter-domain Gaussian processes for sparse inference using inducing features," in *Advances in Neural Information Processing Systems 22*. Cambridge, MA: MIT Press, 2010, pp. 1087–1095.

[20] M. K. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Proc. 12th Int. Workshop Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[21] D. Gu, "Spatial Gaussian process regression with mobile sensor networks," *IEEE Trans. Neural Networks Learn. Syst.,* vol. 23, pp. 1279–1290, 2012.

[22] E. Snelson, Z. Ghahramani, and C. Rasmussen, "Warped Gaussian processes," in *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press, 2003.

[23] M. Lázaro-Gredilla, "Bayesian warped Gaussian processes," in *Advances in Neural Information Processing Systems 26*. Cambridge, MA: MIT Press, 2013.

[24] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Hoboken, NJ: Wiley, 2010.

[25] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Processing*, vol. 56, no. 7, pp. 2781 –2796, July 2008.

[26] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Processing Mag.*, vol. 28, no. 1, pp. 97–123, Jan. 2011.

[27] S. Van Vaerenbergh, I. Santamaría, and M. Lázaro-Gredilla, "Estimation of the forgetting factor in kernel recursive least squares," in *Proc. IEEE Int. Workshop Machine Learning Signal Processing (MLSP),* Sept. 2012, pp. 1–6.

[28] M. Lázaro-Gredilla, S. Van Vaerenbergh, and I. Santamaría, "A Bayesian approach to tracking with kernel recursive least-squares," in *Proc. IEEE Int. Workshop Machine Learning Signal Processing (MLSP),* Sept. 2011, pp. 1–6.

[29] A. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley-IEEE Press, 2003.

[30] K. Feher, *Digital Communications: Satellite/Earth Station Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[31] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 23, no. 1, pp. 22–32, Jan. 2012.

[32] J. Gutiérrez, Ó. González, J. Pérez, D. Ramírez, L. Vielva, J. Ibáñez, and I. Santamaría, "Frequency-domain methodology for measuring MIMO channels using a generic test bed," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 3, pp. 827–838, Mar. 2011.

[33] T. Minka, "Expectation propagation for approximate Bayesian inference," in *Proc. 17th Conf. Uncertainty Artificial Intelligence*, Seattle, WA, 2001, pp. 362–369.

[34] P. Olmos, J. Murillo-Fuentes, and F. Pérez-Cruz, "Joint nonlinear channel equalization and soft LDPC decoding with Gaussian processes," *IEEE Trans. Signal Processing*, vol. 58, no. 3, pp. 1183–1192, 2010.

[35] J. Wang, D. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, Feb. 2008.

[36] A. Liutkus, R. Badeau, and G. Richard, "Gaussian processes for underdetermined source separation," *IEEE Trans. Signal Processing*, vol. 59, no. 7, pp. 3155–3167, July 2011.

[37] L. Pasolli, F. Melgani, and E. Blanzieri, "Gaussian process regression for estimating chlorophyll concentration in subsurface waters from remote sensing data," *IEEE Geosci. Remote Sensing Lett.*, vol. 7, no. 3, pp. 464–468, Mar. 2010.

[38] C. Archambeau, D. Cornford, M. Opper, and J. Shawe-Taylor, "Gaussian process approximations of stochastic differential equations," *J. Mach. Learn. Res.,* vol. 1, pp. 1–16, Mar. 2007.

[39] J. J. Murillo-Fuentes and F. Pérez-Cruz, "Gaussian process regressors for multiuser detection in DS-CDMA systems," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2339–2347, Aug. 2009.

[SP]