Gaussian Processes and its Application to the design of Digital Communication Receivers

Pablo M. Olmos, Juan José Murillo-Fuentes and Fernando Pérez-Cruz

University of Seville, Signal Theory and Communications Department. Spain.
University of Princeton, Princeton (NJ), Electrical Engineering Department. USA.
University Carlos III in Madrid, Department of Signal Theory and Communications. Leganes (Madrid), Spain

Abstract
In this chapter, we introduce Gaussian processes for machine learning and their application to designing digital communication receivers. Gaussian processes for machine learning are Bayesian nonlinear tools for solving regression and classification problems. Gaussian processes for regression (GPR) were introduced in the mid-nineties to solve nonparametric estimation problems from a Bayesian perspective. They place a Gaussian process (GP) prior over the possible regressors and use the available data to obtain a posterior regressor, which it is able to explain the observations without overfitting. The covariance matrix of the GP prior describes the different solutions that can be achieved, e.g. linear, polynomial, or universal regressors. The solution of GPR is analytical given its covariance function and, besides providing point estimates, it also assigns confidence intervals for the predictions. Furthermore, the covariance function can be optimized by maximum likelihood to better represent the data, which adds additional flexibility to our regression approximation.
GPR can be generalized to solve classification problems, namely Gaussian processes for classification (GPC). GPC extends the idea of GPR for a classification likelihood model. For this likelihood, the GPC posterior is no longer analytically tractable and we need to approximate it. Expectation Propagation (EP), which matches the mean and covariance of the GP posterior to a Gaussian distribution, is the most widely used approximation. Unlike most state-of-the-art classifiers, GPC does not return point-wise decisions, but it provides an accurate posterior probability for each classification decision. This is a major advantage to be exploited by subsequent applications for reducing the base error produced by our nonlinear classifiers.
Nonlinear regression and classification techniques have been widely used for designing digital communication receivers for nonlinear channels or whenever there is little information about the channel model or for nonlinear model. These nonlinear tools must use short training sequences to learn the channel and to adapt to a wide range of scenarios, from linear minimum phase to nonlinear and non-minimum phase and from single to multi-user
scenarios. In this framework, Gaussian processes for machine learning can be used instead of other nonlinear tools, such as neural networks or support vector machines, providing several advantages to these widespread techniques. First, their structure can be learnt by maximum likelihood. Hence, we avoid cross-validation techniques; which reduces the number of training samples needed to provide accurate predictions. And, at the same time, we may learn more parameters compared to other state-of-the-art techniques, i.e., we have more flexible models that can easily resort from linear to intricate nonlinear solutions. Second, they provide accurate posterior probability estimates that can be exploited by the channel decoder to reduce the overall error rate of our communication system.

We analytically study how Gaussian processes for machine learning can replace other nonlinear techniques for designing the all-important digital communication receiver. We also present some covariance matrices suitable for general digital communication channels. We illustrate our theoretical results by showing how GPR provides accurate solutions to the channel equalization and multi-user detection problems with very short training sequences and how a low-density parity-check (LDPC) channel decoder might benefit from a GPC equalizer that provides accurate posterior probability estimates.

1. Introduction

Gaussian processes are typically used to characterize the noise component in digital communication systems, as it is mainly caused by thermal noise fluctuations (Salehi & Proakis, 2002). In this chapter, we propose the Gaussian processes (GPs) framework to design nonlinear receivers in digital communication systems. GPs were initially presented as a nonlinear estimation technique in 1978 (O’Hagan & Kingman, 1978) and were rapidly forgotten due to its computation complexity. In the mid-nineties, they were independently rediscovered (Williams & Rasmussen, 1996). Since then, they have been shown to fit many different applications (Williams & Rasmussen, 2006) and nowadays their computational complexity is no longer a limiting issue (Quiñonero-Candela & Rasmussen, 2005).

There is a vast literature on machine learning techniques for designing digital communication systems. The channel equalization problem has been addressed with different machine learning tools, such as multilayered perceptrons (MLPs) (Gibson & Cowan, 1991), radial basis function networks (RBFNs) (Chen et al., 1991), recurrent RBFNs (Cid-Sueiro et al., 1994), self-organizing feature maps (SOFMs) (Kohonen et al., 1991), wavelet neural networks (Chang & Wang, 1995), GCMAC (González-Serrano et al., 1998), kernel adaline (KA) (Mitchinson & Harrison, 2002), or support vector machines (SVMs) (Pérez-Cruz et al., 2001), among many others. Other digital communication systems that have also benefited from nonlinear detection and estimation algorithms are multiuser detection (Cruickshank, 1996), (Tanner & Cruickshank, 1997), (Caffery & Stuber, 2000), and (Pham et al., 2007), multiple-input multiple-output systems (Sánchez-Fernández et al., 2004), beamforming (Martinez-Ramón et al., 2007), predistortion (González-Serrano et al., 2001), and plant identification (Arenas-Garcia et al., 2006) to name a few.

For these machine learning approaches, it is necessary to prespecify the hyperparameters (structure), since standard methods for searching the optimal hyperparameters, i.e., cross validation (Kimeldorf, G.S. & Wahba, 1971), (Bishop, 1995) require immense computational resources, which are not available in most communication receivers, and also their training
time is highly variable. As a result, they use a suboptimal structure that requires longer training sequences for ensuring optimal receiver performance.

Gaussian processes for machine learning are rooted in Bayesian statistics (Williams & Rasmussen, 2006), and consequently allow building a likelihood function for its hyperparameters given the training examples. This likelihood can be optimized to set the hyperparameters. This property makes GPs an attractive tool for designing nonlinear digital communication receivers, compared to other nonlinear machine learning tools, because the hyperparameters can be optimally set for each instantiation of our problem with a single optimization procedure. For short training sequences, hyperparameter mismatch significantly affects the performance of digital communication receivers, while for longer training sequences, this performance is not sensitive to variations in the hyperparameters. Most papers applying nonlinear machine learning for designing digital communication receivers propose fixed hyperparameters and sufficiently long training sequences. Our proposal is to introduce GPR with optimally trained hyperparameters to reduce the length of the training data sequence. We experimentally illustrate that previous fixed hyperparameters machine learning tools clearly underperform the GPR. In addition, we show that GPR can be understood as a nonlinear MMSE estimator. Therefore GPR achieves optimal results from the MMSE viewpoint, a quite extended criterion in digital communications.

Gaussian processes can be extended for solving classification problems, namely GPC. In this case, the posterior is no longer tractable and we need to use approximations to compute the prediction for each class label (Williams & Rasmussen, 2006). A Gaussian distribution is typically used to approximate the GPC posterior, either using Laplace (Williams & Barber, 1998) or Expectation Propagation methods (Kuss & Rasmussen, 2005), because the posterior is unimodal for a large enough training set. GPC is a Bayesian tool that provides the posterior probability for each decision. This property makes it unique among the nonlinear methods for channel equalization. A soft equalizer allows the channel decoder to significantly reduce its error rate, because it has individual information about which bits might be in error. GPC outperforms the SVM with a probabilistic output (Platt, 2000). This method passes the SVM output through a sigmoid, which is not quite principled, as Platt himself explains in (Platt, 2000), but it typically provides good predictions. However, in some cases, its probability predictions are not accurate, as shown in (Kuss & Rasmussen, 2005), because we cannot know a priori how good this fit would be.

The GPR can also be used to design linear and nonlinear multiuser detectors (MUDs) in CDMA communications. The GPR solution for MUD receivers is advantageous in several ways, when compared to other nonlinear machine learning tools (e.g. MLPs, RBFNs and SVMs). First, the GPR solution is analytical, given its parameters. There is no need for solving a complex optimization problem at this stage. Second, given a training dataset, a likelihood function for the GPR parameters can be stated and, hence, its parameters can be optimally set (in maximum likelihood sense). MLPs, RBFNs or SVMs need to specify beforehand its structure or parameters, which can be suboptimal for each individual instantiation of our problem. These characteristics translate to shorter training sequences and improved convergence for GPR-based CMDA receivers.

The rest of the chapter is organized as follows. In Section 0 we describe the equalization and multi-user detection problems and we introduce the channel model considered through this chapter. We present the design of digital communication receivers as an optimization problem in Section 0 and show how different nonlinear machine learning tools can be fitted
2. Equalization and multi-user detection.

2.1 Channel model
We consider throughout the chapter the following deterministic channel model:

\[ x = Hs + z, \]

where \( s \) is a random variable column-vector representing the transmitted symbols, \( H \) corresponds to the deterministic channel gains, unknown to both the transmitter and receiver, \( z \) is zero-mean Gaussian noise, and \( x \) represents the received symbols. This model is general enough to capture most standard communication systems. We can also combine different \( H \) matrices to accommodate other communication systems.

I. **Intersymbol interference**: each element in \( s \) is a symbol transmitted at a different time instant. \( H \) is a row vector that represents the channel impulsive response.

II. **Multiple-input multiple-output**: \( H_{ij} \) represents the gain from the \( i \)-th receiving antenna to the \( j \)-th transmitting antenna and \( s \) represents the symbols transmitted by the antenna array.

III. **Fading**: \( H \) is a diagonal matrix with the fading coefficients and \( s \) represents the symbols transmitted at each time instant.

IV. **CDMA**: the columns of \( H \) collect each user’s spreading code and each element of \( s \) represents the symbol transmitted by the user.

2.2 Equalization in digital communication receivers
In Fig. 1, we depict a simple band-based model to describe a nonlinear dispersive Single Input-Single Output (SISO) communication channel. In this case, \( H \) is a row vector representing the coefficients of the impulse response of the channel:

\[ h(z) = \sum_{i=0}^{n_c-1} h_i z^{-i}, \]

\[ H = \begin{bmatrix} h_0 & h_1 & \cdots & h_{n_c} \end{bmatrix}, \]

where \( n_c \) denotes the channel length. The nonlinearity in the channel, due to amplifiers and converters in the receiver can be modeled as (Mitchinson & Harrison, 2002):

\[ x[j] = g(r[j]) + z[j] = g(Hs_j) + z[j], \]

where \( g \) is a nonlinear function the inputs face at the receiver (amplifiers and converters) and \( r \) is a random variable column-vector representing the transmitted symbols.
where \( g(\cdot) \) is the nonlinear function the inputs face at the receiver (amplifiers and converters) and \( s_j \) is a column vector where each element is a symbol transmitted at a different time instant:

\[
s_j = \left[ s[j] \ s[j-1] \ \cdots \ s[j-n_C+1] \right]^T. \tag{4}
\]

The equalizer collects a set of \( n_{EQ} \) consecutive received symbols:

\[
x_j^{EQ} = [x[j+\tau], x[j-1+\tau], \ldots, x[j - n_{EQ} + 1 + \tau]], \tag{5}
\]

to predict each symbol \( s_j = s[j] \), where \( n_{EQ} \) and \( \tau \) represents, respectively, the length and the delay of the equalizer. In Section 0, we propose the use of GPC and GPR for channel equalization.

![Fig. 1. Simple discrete-time SISO transmission channel model.](image)

**2.3 Multi-user detection in CDMA communications**

In this chapter we focus on synchronous DS-CDMA, (Verdú, 1998), and we assume all users transmit at the same symbol rate using a BPSK modulation. The obtained results can be generalized to other scenarios, as asynchronous DS-CDMA, different rates, or modulations. The discrete chip-rate-sampled baseband synchronous DS-CDMA model proposed in (Verdú, 1998) is illustrated in Fig. 2. In this model, \( k \) symbols are transmitted (one per user) at instant \( j \):

\[
u_j = [s_j(1), s_j(2), \ldots, s_j(k)]^T. \tag{6}
\]

Each user’s symbol, \( s_j(l) \), is multiplied by its spreading code, \( c_l \), which is a sequence of \( n_S \) pseudorandom binary values regarded as chips. The resulting signal is amplified by a different gain, \( a_l \), i.e., in the downlink of a mobile (cellular) communication system larger amplitudes are assigned to users further away, causing the near-far problem to users closest to the base station. The \( n_S \)-chip signal at the receiver end yields:
where $H_S$ is an $n_S \times k$ matrix whose columns contain the spreading codes, $A$ is a $k \times k$ diagonal matrix containing the user amplitudes and $z$ is an $n_S M$ dimensional column-vector with additive white Gaussian noise (AWGN) of variance $\sigma_z^2 I$.

$$x_{mud}^j = H_m \begin{bmatrix} H_S A & 0 & \ldots & 0 \\ 0 & H_S A & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & H_S A \end{bmatrix} \begin{bmatrix} u_j \\ u_{j-1} \\ \vdots \\ u_{j-M+1} \end{bmatrix} + z = H v_j + z, \quad (7)$$

Fig. 2. System model for DS-CDMA.

We have pre-multiplied the received chips by $H_m$ to incorporate the effect of a multipath channel in the CDMA system (Chen et al., 2001). $H_m$ is a Toeplitz matrix, in which each row represents the channel impulsive response. We consider a time-invariant channel with inter-symbolic interference (ISI) characterized by its discrete channel impulsive response $h(z)$ of length $n_C$ in chip periods. The length of the channel response times the chip period is the maximum delay considered in our multipath model. The matrix $H$ in (7) summarizes the effect of the channel, the spreading codes and the different amplitudes for each user, where $v_j = \left[ u_j^T \quad u_{j-1}^T \quad \ldots \quad u_{j-M+1}^T \right]^T$ is a $kM$ dimensional vector including the transmitted bits.

The objective for the DS-CDMA multi-user detector (MUD) receiver is to recover the transmitted bit for a particular user, the UoI. Linear MUDs are useful when the ISI is negligible and the codes are quasi-orthogonal. When the multipath effect and the near-far problem are strong, the optimal detector becomes highly nonlinear. The nonlinearity of the detector is significantly more disruptive for short spreading codes. In these scenarios nonlinear detectors are useful. Nonlinear MUDs for DS-CDMA estimate the symbol of the UoI as $\tilde{s}_j(l) = f(x_j)$. If we knew all the $2^{kM}$ possible received noise free states, we could derive a MUD by studying a Bayes-optimal classifier (Chen et al., 2001). This optimal one-shot detector is given by:
\[ \hat{s}_j(l) = \text{sign} \left( \sum_{i=1}^{2^{LM}} \frac{s^{(i)}_l}{\sqrt{2\pi\sigma_z}} \exp \left( -\frac{||x_j^{\text{mud}} - Hv^{(i)}||^2}{2\sigma_z^2} \right) \right), \]

where \( s^{(i)}_l = \{ \pm 1 \} \) is the class label for the \( i \)-th noise-free state \( Hv^{(i)} \). This structure resembles that of the Gaussian used in (Mitra & Poor, 1994), and suggests Gaussian kernels in SVM, as proposed in (Chen et al., 2001). But its complexity is exponential in the number of users and the length of the channel response, provided we have the \( 2^{LM} \) possible free states.

3. Nonlinear optimization for communication receivers

3.1 Minimum mean squared error detector

Given the channel model in (1), we can estimate, in the receiver, the transmitted vector using a minimum mean squared error (MMSE) detector (Kay, 1993):

\[ f_{\text{mmse}}(x) = \arg\min_{f(\cdot)} E \left[ \left\| s - f(x) \right\|^2 \right]. \]

The function \( f_{\text{mmse}}(x) \) is the mean value of \( s \) given the received vector \( x \), \( E[s \mid x] \), which is a linear function of \( x \) if \( s \) is a Gaussian distribution. Practical structural constraints dictate the use of discrete constellations, such as PSK and QAM, which depart from the optimal Gaussian distributions. Although linear detectors cannot achieve \( E[s \mid x] \) if \( s \) is a discrete random variable, and thus the MMSE is only a proxy for minimizing the probability of misclassification. Still digital communication receivers use linear MMSE detectors for estimating the transmitted vector, because they can be easily implemented and hopefully their performance is not severely degraded. The linear MMSE, \( f(x) = w^T x \), detector can be expressed as a Wiener filter:

\[ w_{\text{mmse}} = \arg\min_w E \left[ (s - w^T x)^2 \right] = \left( E[xx^T] \right)^{-1} E[xs]. \]

3.2 Machine learning for digital communication receivers

The design of digital communication receivers can be readily understood as a supervised classification problem (Gibson et al, 1991) in which the receiver constructs a classifier for deciding over the incoming symbols. Machine learning tools optimize the risk of misclassification:

\[ f_{\text{opt}}(x) = \arg\min_{f(\cdot)} E \left[ L(s, f(x)) \right] = \arg\min_{f(\cdot)} \int L(s, f(x)) p(s, x) ds dx, \]
where $L(\cdot)$ is a loss function that measures the penalty for wrongly classifying a pattern, and $f(x)$ is the nonlinear model to predict $s$.

The joint density, $p(s,x)$, is typically unknown and thus we use a training sequence $\{x_i,s_i\}_{i=1}^{n}$ and the empirical risk minimization (ERM) inductive principle (Vapnik, 1998) to obtain the optimal solution:

$$\hat{f}_{opt}(x) = \underset{f(\cdot)}{\arg \min} \left\{ \sum_{i=1}^{n} L(s_i, f(x_i)) + \lambda \Omega(||f||) \right\},$$  \hspace{1cm} (12)

where we have included a regularization term, $\lambda \Omega(||f||)$, to avoid overfitting and to ensure that the minimum of the empirical risk converges to the minimum risk (Vapnik, 1998) as the number of training samples increases.

If we choose $f(x) = w^T \phi(x)$, $L(s, f(x)) = (s - w^T \phi(x))^2$ and $\Omega(f) = ||w||^2$ we get a convex functional,

$$w_{nl_{\text{mmse}}} = \underset{w}{\arg \min} \left\{ \sum_{i=1}^{n} (s_i - w^T \phi(x_i))^2 + \lambda ||w||^2 \right\},$$  \hspace{1cm} (13)

which can be analytically solved,

$$w_{nl_{\text{mmse}}} = \left( \Phi^T \Phi + \lambda I \right)^{-1} \Phi^T s,$$  \hspace{1cm} (14)

where $\Phi = [\phi(x_1), \ldots, \phi(x_n)]^T$ and $s = [s_1, \ldots, s_n]^T$. We denote this solution as nonlinear MMSE, since it is a nonlinear extension of (10) in which we have replaced $x$ by $\phi(x)$ and the expectations by sample averages.

In the next section we show (14) is equivalent to the mean solution provided by Gaussian processes for regression with a Gaussian likelihood function and that it can be solved using kernels (Pérez-Cruz & Bousquet, 2004).

### 4. Gaussian Processes for Regression

In a nutshell, Gaussian processes for regression (GPR) assume that a GP prior governs the set of possible regressors. Consequently, the joint distribution of training and test data is given by a multidimensional Gaussian density function, and the predicted distribution for each test point is estimated by conditioning on the training data.

We mainly present GPR from the Bayesian generalized linear regression viewpoint. Although from this opening we lose the GPs interpretation. We believe it is a simpler way to understand GPR. This approach mimics how most machine learning textbooks introduce
nonlinear regression (Bishop, 1995), (Schölkopf & Smola, 2001) and (Haykin, 1999) and it helps understanding GPR as a nonlinear MMSE estimation. Therefore, practitioners in signal processing for digital communications can readily relate to this new tool for estimation and detection. Another point of view is that of the function space. Both interpretations are described in (Williams, 1999), where they are shown to be identical for Gaussian likelihood models. There is more about GPs than what we introduce in this summary, for interested readers, GPs extensions can be found in (Williams & Rasmussen, 2006).

4.1 Weight-space view
A generalized linear regressor expresses the input-output relation as

\[ s = \mathbf{w}^\top \phi(\mathbf{x}) + \nu, \]  

(15)

where \( \phi(\cdot) \) is a nonlinear transformation to a higher dimensional feature space and \( \nu \) is a random variable that measures the deviation between \( s \) and its estimate. In Bayesian machine learning, given a labeled training sequence, \( \mathcal{D} = \{\mathbf{x}_i, s_i\}_{i=1}^n \) where the input \( \mathbf{x}_i \in \mathbb{R}^d \) and the output \( s_i \in \mathbb{R} \), and a statistical model for \( \nu \), \( \mathbf{w} \) is considered to be a random variable and, to predict the outcome of \( \mathbf{x}_* \), we use its conditional density given the training data set, \( p(\mathbf{w} \mid \mathcal{D}) \). This conditional density, known as the posterior of \( \mathbf{w} \), can be computed through Bayes rule,

\[
p(\mathbf{w} \mid \mathcal{D}) = p(\mathbf{w} \mid \mathbf{s}, \mathbf{X}) = \frac{p(\mathbf{s} \mid \mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{s} \mid \mathbf{X})} = \frac{p(\mathbf{w})}{p(\mathbf{s} \mid \mathbf{X})} \prod_{i=1}^n p(s_i \mid \mathbf{x}_i, \mathbf{w}),
\]

(16)

where \( p(s_i \mid \mathbf{x}_i, \mathbf{w}) \) is the likelihood function of \( \mathbf{w} \), \( p(\mathbf{w}) \) its prior distribution, \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \) and \( p(\mathbf{s} \mid \mathbf{X}) \) is the evidence of the model. To predict the output for a new test point \( \mathbf{x}_* \) we integrate out \( \mathbf{w} \) :

\[
p(s_* \mid \mathbf{x}_*, \mathcal{D}) = \int p(s_* \mid \mathbf{x}_*, \mathcal{D}, \mathbf{w})p(\mathbf{w} \mid \mathcal{D})d\mathbf{w},
\]

(17)

in which the conditional density of each \( s_* \) (the likelihood of \( \mathbf{w} \)) is weighted by the posterior of \( \mathbf{w} \) and is summed over all possible \( \mathbf{w} \). As a result, we get a full statistical description of \( s_* \), given all the available information (\( \mathbf{x}_* \) and \( \mathcal{D} \)). In this setting, we predict the value of \( s_* \) using the full statistical model of \( \mathbf{w} \), not only its maximum likelihood estimate. This setting is
quite general, as we can use any model for the likelihood and prior for solving the regression estimation problem. Gaussian likelihood, \( p(s \mid x, w) = \mathcal{N}(w^\top \phi(x), \sigma_w^2) \), leads to the MMSE criterion; and a zero-mean Gaussian prior, \( p(w) = \mathcal{N}(0, \sigma_w^2 I) \), allocates probability mass to every possible \( w \) and allows solving (17) analytically. The posterior distribution in (16) is then a Gaussian density function, \( p(w \mid D) = \mathcal{N}(\mu_w, \Sigma_w) \), where

\[
\mu_w = \sigma_w^2 \left( \sigma_w^2 \Phi^\top \Phi + \sigma_p^2 I \right)^{-1} \Phi^\top s,
\]

\[
\Sigma_w^{-1} = \frac{1}{\sigma_p^2} \Phi^\top \Phi + \frac{1}{\sigma_w^2} I.
\]

Actually, the posterior mean in (18) is identical to the maximum a posteriori (MAP) of (16):

\[
\mu_w = w_{MAP} = \arg \max_w \left\{ p(w \mid s, X) \right\}
\]

\[
= \arg \max_w \left\{ \log p(s \mid X, w) + \log p(w) \right\}
\]

\[
= \arg \max_w \left\{ -\frac{1}{\sigma_p^2} \sum_{i=1}^n (s_i - w^\top \phi(x_i))^2 - \frac{1}{\sigma_w^2} \| w \|^2 \right\},
\]

which is identical to (12) for \( \lambda = \sigma_p^2 / \sigma_w^2 \). We can also check that (18) is equal to (14) and, therefore the GPR mean prediction can be regarded as a nonlinear MMSE estimation for the nonlinear mapping \( \phi(\cdot) \). The prediction for \( s_* \) in (17) is a Gaussian density function \( p(s_* \mid x_*, D) = \mathcal{N}(\mu_{s_*}, \sigma_{s_*}) \):

\[
\mu_{s_*} = \phi^\top(x_*) \mu_w = \phi^\top(x_*) \frac{1}{\sigma_w^2} \Sigma_w \Phi^\top s,
\]

\[
\sigma_{s_*}^2 = \phi^\top(x_*) \Sigma_w \phi(x_*) = \phi^\top(x_*) \left( \frac{1}{\sigma_p^2} \Phi^\top \Phi + \frac{1}{\sigma_w^2} I \right)^{-1} \phi(x_*).
\]

### 4.2 Alternative formulation

There is an alternative formulation for \( \mu_{s_*} \) and \( \sigma_{s_*}^2 \), in which we do not need to know the nonlinear mapping \( \phi(\cdot) \) and we only need to work with its inner product or kernel, defined as:
$k(x_i, x_j) = \sigma_w^2 \phi^\top(x_i)\phi(x_j). \quad (23)$

To obtain this alternative formulation, we first define the covariance matrix $C$ as

$$(C)_{ij} = k(x_i, x_j) + \sigma_n^2 \delta_{ij}, \quad (24)$$

which can be related to $\Sigma_w$ as follows:

$$\Sigma_w^{-1}\Phi^\top = \left( \frac{1}{\sigma_w^2} \Phi^\top \Phi + \frac{1}{\sigma_n^2} \mathbf{I} \right)^\top \Phi^\top = \frac{1}{\sigma_w^2 \sigma_n^2} \Phi^\top C. \quad (25)$$

Now if we premultiply (25) by $\Sigma_w$ and postmultiply it by $C^{-1}$, we obtain the following equivalency: $\Sigma_w \Phi^\top / \sigma_n^2 = \sigma_w^2 \Phi^\top C^{-1}$, which can be used to simplify (22) and express the GPR prediction mean as

$$\mu_s = \phi^\top(x_s)\sigma_w^2 \Phi^\top C^{-1}s = k^\top C^{-1}s, \quad (26)$$

where

$$k = \sigma_w^2 \phi^\top(x_s) \Phi^\top = [k(x_s, x_1), \ldots, k(x_s, x_n)]^\top. \quad (27)$$

To compute the prediction for any vector $x_s$, we do not need to know the nonlinear mapping $\phi(\cdot)$, only its kernel. The complexity of computing $\mu_s$ in (26) is linear in the number of training examples, because we can pre-compute the vector $C^{-1}s$ that does not depend on $x_s$ and we only need to filter $k$ with it for each new test pattern.

We can also define the variance of our predictor using kernels as follows:

$$\sigma_s^2 = k(x_s, x_s) - k^\top C^{-1}k, \quad (28)$$

which is achieved after applying to (19) the matrix inversion lemma (Scharf, 1990). Equations in (26) and (28) represent the predictions for $x_s$ given by the Gaussian processes view of GPR. The matrix $C$ is the covariance matrix of a multidimensional Gaussian distribution, hence its name, that describes the training data, and the vector $k$ represents the covariance vector between the training dataset and the test vector. Therefore, the function
4.3 Function-space view

An equivalent and alternative way of reaching similar results is to consider inference directly in the so-called function space, where we use a Gaussian process to describe a distribution over functions. If \( f(x) = w^T \phi(x) \) we may rewrite (17) as

\[
p(s_* | x_*, D) = \int p(s_* | x_*, D, f_*) p(f_* | x_*, D)df_*,
\]

where \( f_* = f(x_*) \) and proceed in a similar way to estimate (16) and solve it for a Gaussian prior on \( f(x) \) and Gaussian likelihood \( p(s | x, f) = \mathcal{N}(f(x), \sigma_s^2) \):

\[
p(f_* | x_*, D) = \int p(f_* | x_*, x, f)p(f | D)df,
\]

where

\[
p(f | D) = p(f | X, s) = \frac{\prod_i p(s_i | f_i)p(f | X)}{p(s | X)}.
\]

5. Gaussian processes for classification

Gaussian process for classification is a bit trickier than the regression counterpart, because we cannot rely on a Gaussian likelihood function to predict the labels of each class as the outcomes come from a discrete set (Williams & Rasmussen, 2006). Thereby to predict the class labels we need to resort to numerical integration or approximations to tractable density models. A generalized linear binary classifier predicts the class label for an input \( x \) as follows:

\[
p(s = +1 | w, x) = p(s = +1 | f(x)) = \sigma(f(x)),
\]

where \( f(x) = w^T \phi(x) \) is an underlying continuous function, \( \sigma(\cdot) \) is a sigmoid that squashes \( f(x) \) between 0 and 1, and \( p(s = -1 | f(x)) = 1 - p(s = +1 | f(x)) \). Function \( \sigma(\cdot) \) is typically the logistic function or the cumulative density function of a Gaussian (Williams & Rasmussen, 2006).

Given a labeled training sequence \( D = \{x_i, s_i\}_{i=1}^n \), where the input \( x_i \in \mathbb{R}^d \) and the output \( s_i \in \{\pm 1\} \), we can compute the posterior over the underlying function.
\[ f = [f(x_1), \ldots, f(x_n)]^T \] using Bayes rule, as we did in the last section for GPR with \( w \), and we can integrate out \( f \) to predict the class label for any new test point \( x_* \). We can compute the class label for the test samples as follows:

\[
p(s_* = +1 \mid x_*, \mathcal{D}) = \int \sigma(f_*)p(f_* \mid x_*, \mathcal{D})df_*. \tag{33}
\]

Now, equations (30) and (31) are intractable due to the likelihood model employed for \( f(x) \) in (32). GPC typically relies on a Gaussian approximation for the posterior density \( p(f \mid \mathcal{D}) \), to analytically solve (30), and (33) is a one-dimensional integral that can be easily solved numerically. The standard approximations to the posterior \( p(f \mid \mathcal{D}) \) are Laplace or Expectation Propagation, as explained in (Kuss & Rasmussen, 2005). Further details on how to approximate the posterior can be found in (Williams & Rasmussen, 2006).

6. Hyperparameter optimization

If either \( \phi(\cdot) \) or \( k(\cdot, \cdot) \) are known, we can predict the output of any incoming sample in a regression (26) or a classification problem (33). But for most estimation problems, the best nonlinear transformation (or its kernel) is unknown. Therefore, it is usually defined in a parametric form as function of the so-called hyperparameters. The optimal setting of the hyperparameters could be obtained by cross-validation, similarly to any other nonlinear machine learning method. From the point of view of Bayesian machine learning, we can proceed as we did for the parameters \( w \) in Section 0. First, we compute the likelihood of the hyperparameters of the kernel given the training dataset:

\[
p(s \mid X, \theta) = \int p(s \mid f, X, \theta)p(f \mid X, \theta)df, \tag{34}
\]

where \( \theta \) represents the hyperparameters of the covariance function or kernel. We have added to explicitly indicate the dependence on the kernel’s hyperparameters. This was omitted in the GPR and GPC presentations in Sections 0 and 0 for clarity purposes.

Second, we can define a prior for the hyperparameters, \( p(\theta) \), that can be used to construct its posterior:

\[
p(\theta \mid \mathcal{D}) = \frac{p(s \mid X, \theta)p(\theta)}{p(s \mid X)}, \tag{35}
\]

Third, we can integrate out the hyperparameters to obtain the predictions:

\[
p(s_* \mid x_*, \mathcal{D}) = \int p(s_* \mid x_*, \mathcal{D}, \theta)p(\theta \mid \mathcal{D})d\theta. \tag{36}
\]
However, in this case, the likelihood of the hyperparameters does not have a conjugate prior and the posterior is non-analytical. Hence the integration has to be done either by sampling or approximations. Although this approach is well principled, it is computationally intensive and it demands a high number of samples. Thereby it is not feasible for digital communications receivers.

Alternatively, we can use the likelihood function in (34) of the hyperparameters and compute its maximum to obtain its optimal setting (Williams & Rasmussen, 1996), which is used to describe the kernel for the test samples. Although setting the hyperparameters by maximum likelihood is not a purely Bayesian solution, it is fairly standard in the community and it allows using Bayesian solutions in time-sensitive applications. The maximum likelihood hyperparameters are given by

$$\theta_{ML} = \arg \max_{\theta} p(s \mid X, \theta).$$

(37)

This optimization is nonconvex (Mackay, 2003). But as we increase the number of training samples, the likelihood becomes a unimodal distribution around the maximum likelihood hyperparameters and the ML solution can be found using gradient ascent techniques. See (Williams & Rasmussen, 1996) for further details.

### 6.1 Covariance matrix

To optimize the kernel hyperparameters in (37), we need to describe a kernel in a parametric form. Kernel design is one of the most challenging open problems in machine learning, as it is mainly driven by each particular application. We need to incorporate our prior knowledge into the kernel, but, at the same time, we want the kernel to be flexible to explain previously unknown trends in the data. In (Williams & Rasmussen, 1996), a list of flexible kernels (e.g., linear, Gaussian, neural networks, Matérn, among others) is presented along with their properties.

For example, if we know the optimal solution to be linear, we could use the linear kernel:

$$k(x, x') = \sigma_w^2 x^\top x'.$$

In general, kernel functions are more complex and they incorporate several hyperparameters. For example, the Gaussian kernel with automatic relevance determination (ARD) proposes one nonnegative weight, $\gamma_{\ell}$, per input dimension:

$$k(x_i, x_j) = \alpha_1 \exp \left\{ -\sum_{\ell=1}^{d} \gamma_{\ell} \left( x_{i\ell} - x_{j\ell} \right)^2 \right\} + \alpha_2 x_i^\top x_j + \alpha_0 \delta_{ij},$$

(38)

where we have added a linear kernel to use this covariance function for designing digital communication receivers. For this kernel function we define the hyperparameters as $\theta = [\log \alpha_0, \log \alpha_1, \log \alpha_2, \log \gamma_1, \cdots, \log \gamma_d]$, because these hyperparameters need to be positive to ensure that $k(\cdot, \cdot)$ is a positive semi-definite function. Hence, we can apply unconstrained optimization tools when we optimize over $\theta$. The covariance function in (38) is a good kernel for designing digital communication receivers using GPR and GPC, because it contains a linear and a universal nonlinear part.
The linear part can mimic the best linear decision boundary and the nonlinear part modifies it, where the linear explanation is not optimal to obtain the expectation of $s$ given $x$. If the channel is linear, then the ML solution sets $\alpha_1 = 0$ and there is no interference of the nonlinear term with the linear one in the solution. Also, using Gaussian kernel, also denoted as radial basis function (RBF) kernel, for the nonlinear part seems an appropriate choice to achieve nonlinear decisions for digital communication receivers, because the received symbols form a constellation of clouds of points with Gaussian spread around its centers. Due to the symmetry in communication problems and to avoid overfitting, we use the same length scale for all dimensions: $\gamma = \gamma_1 = \cdots = \gamma_d$.

7. Equalization with Gaussian processes.

We return to the system model described in Fig. 1. In this section, we face the experimental study of the Gaussian processes based equalizer. We can apply either regression (GPR) (Pérez-Cruz et al., 2008) or classification (GPC) (Pérez-Cruz et al., 2007). The GPR solution has the advantage of being analytical, which makes it easy to compute, while to obtain the GPC solutions we need to use approximations for its posterior distribution, which can be time consuming. However, the GPR solution cannot be interpreted as posterior probabilities for each output, because it assumes it is solving a regression problem with Gaussian noise. And that is not the case for channel equalization, which can be cast as a classification problem.

The GPR input is a set of $n_{EQ}$ consecutive received symbols. It is first trained with a set $\mathcal{D} = \{\mathbf{x}_{i}^{EQ}, s_i\}_{i=1}^{n}$, where $\mathbf{x}_{i}^{EQ} \in \mathbb{R}^{n_{EQ}}$. The solution of the GPR process for a new input $\mathbf{x}_* \in \mathbb{R}^{n_{EQ}}$ is the posterior mean $\mu_{s_*}$ (26) and variance $\sigma_{s_*}^2$ (28). The decision about the $s_*$ symbol is based on the $\mu_{s_*}$ parameter. In the binary case, $s \in \{ \pm 1 \}$, the GPR-equalizer decides which the transmitted bit was according to:

$$
\hat{s}_* = \text{sign}(\mu_{s_*}) = \text{sign}(k^\top \mathbf{C}^{-1}s).
$$

A GPC equalizer is also trained with a training set $\mathcal{D} = \{\mathbf{x}_{i}^{EQ}, s_i\}_{i=1}^{n}$, to predict the probability $p(s_* = +1 \mid \mathbf{x}_i^{EQ}, \mathcal{D})$ for a new input $\mathbf{x}_i^{EQ}$. The symbol $s_*$ can be estimated by

$$
\hat{s}_* = \begin{cases} +1 & p(s_* = +1 \mid \mathbf{x}_i^{EQ}, \mathcal{D}) > 0.5 \\ -1 & p(s_* = +1 \mid \mathbf{x}_i^{EQ}, \mathcal{D}) < 0.5 \end{cases}.
$$

In the next section, we show that the Bit Error Rate (BER) performance obtained with GPR and GPC equalizers in (39) and (40), is similar. If the information bits $m[j]$, see Fig. 3, are encoded into a binary sequence $s[j]$ using a channel code, the resulting BER will be further
reduced if the channel decoder takes as input the GPC posterior probability estimates \( p(s_x = +1 \mid x_s, D) \) instead hard decisions using (39) or (40). Any state-of-the-art procedure as Low-Density Parity-Check (LDPC) codes or Repeat and Accumulate (RA) codes (Mackay, 2003) will provide BER below \( 10^{-5} \) for \( snr \) close to capacity. In the experimental section, we compare the performance obtained with the receiver in Fig. 3 when both GPC and SVM are used as equalizer. As we mentioned in the introduction, SVM soft-output can be transformed into posterior probability by using the method proposed by Platt (Platt, 2000).

\[
\begin{align*}
\hat{m}[j] & \rightarrow \text{Channel} \\
& \rightarrow \text{Encoder} \\
& \rightarrow s[j] \\
\end{align*}
\]

\[
\begin{align*}
\text{Transmitter} \\
\end{align*}
\]

\[
\begin{align*}
h(z) \rightarrow r[j] \rightarrow g(\cdot) \rightarrow z[j] \\
& \rightarrow x[j] \\
\end{align*}
\]

\[
\begin{align*}
\hat{s}[j] & = \{\pm 1\} \\
& \rightarrow \text{Channel} \\
& \rightarrow \text{Decoder} \\
& \rightarrow \text{Equalizer} \\
& \rightarrow \hat{m}[j] \\
\end{align*}
\]

\[
\begin{align*}
p(s[j] = +1) \\
\end{align*}
\]

Fig. 3. Discrete time channel model, together with the transmitter and receiver proposed, including the channel coder and decoder.

7.1 Experimental results
In the next experiments, we deal with the equalization problem in nonlinear channels. The results in these experiments allow drawing some general conclusions about the advantages of GPs for designing digital communication receivers. The channel model is given by:

\[
h(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}. \quad (41)
\]

We add a memoryless nonlinearity to the receiver that transforms each received signal as follows:

\[
g(r) = r + 0.2r^2 - 0.1r^3. \quad (42)
\]

This channel was proposed in (Mitchinson & Harrison, 2002) for modeling radio communication channels and nonlinear amplifiers in wireless communication receivers. To construct the equalizers, we use six received samples to predict each transmitted symbol with a delay of two samples. For the GPC and GPR equalizers, we use the kernel proposed in (38) and SVM is trained with two different kernels: one of them is a linear kernel and the other one is a Gaussian kernel (Pérez-Cruz & Bousquet, 2004). We use a simpler kernel for SVM, because (38) has far too many parameters that cannot be learned with short training sequences. This topic is discussed in detail in (Pérez-Cruz & Murillo-Quintero, 2008). For the SVMs we train a set of receivers with different hyperparameters and we report the best...
result. Thereby, the comparison is biased in favor of the SVM when compared to the GPR and GPC solutions.

In Fig. 4, we show the BER versus the \( \text{slnr} \) for all equalizers and \( n = 512 \). For \( \text{slnr} \) less than 22 dB, the nonlinear GPR equalizer achieves the minimum BER with a gain larger than 3 dB for BER around \( 10^{-3} \). For larger \( \text{slnr} \), the performance of this nonlinear equalizer degrades and the linear equalizers perform significantly better. The nonlinear SVM equalizer performs as the GPR equalizer for \( \text{slnr} \) lower than 17 dB, but for larger \( \text{slnr} \) the training sequence is not long enough and its solution degrades (overfitting). For \( \text{slnr} \) larger than 20 dB, the nonlinear SVM equalizer is not able to reduce the achieved BER. The nonlinear SVM and the GPR as the \( \text{slnr} \) increases are not able to get optimal equalizers, because there is not enough diversity in the training sequence and they overfit to it. The GPR performance is better than the SVM for large \( \text{slnr} \), because it uses a covariance function in (28) that incorporates a linear term. Although it overfits the nonlinear part, the linear component allows the GPR to reduce the BER for large \( \text{slnr} \). If we had increased the training sequence, the SVM and GPR would perform better than the linear methods for larger values of the \( \text{slnr} \).

![Fig. 4.](image-url)

Fig. 4. We report the BER versus \( \text{slnr} \) for a channel equalization problem with the channel model in (41) and (42). The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash dotted line the GPC, the dotted line with circles the linear SVM, and the dotted line with bullets the nonlinear SVM.

The GPC shuts down the nonlinear part and performs as the linear SVM. They are both able to perform significantly better than the linear MMSE, because the channel model is nonlinear. For a nonlinear channel, the received constellation is no longer symmetric, and penalizing the squared error is suboptimal, as it forces that all the detected symbols to be equally far from its optimal value (Pérez-Cruz & Murillo-Fuentes, 2008). The SVM and GPC equalizers only care, if the points are correctly classified, and they only focus on those that
might not be, which explains the BER gap between the linear MMSE equalizer and the GPC and linear SVM ones. In any case, for the $snr$ between 10 and 20 dB, the GPR receivers (and nonlinear SVM) are significantly better than the linear methods and the GPC.

As mentioned before, the channel decoding process can take advantage of the GPC posterior estimation. The GPR solution cannot be interpreted as posterior probabilities for each output bit, because it assumes it is solving a regression problem with Gaussian noise. We illustrate that the novel approach based on GPC provides accurate predictions for posterior probabilities. We consider a linear channel with a simpler transfer function that will actually allow us to compare the true posterior probability and the one provided by the GPC equalizer:

$$h(z) = 1 + 0.5z^{-1}. \tag{43}$$

We have shown in Fig. 5-Fig. 7, the calibration curves for the GPC equalizer compared to the optimum posterior probabilities for $10^4$ test samples. Training sequences with 100, 200 and 800 samples are considered. In the horizontal axis we report the true posterior probability and in the vertical axis the probability predicted by the GPC equalizer. In these plots we can see that, as we increase the number of training samples, the points concentrate on the diagonal axis, which represents the perfect fit between GPC equalizer and the optimum. Hence, the longer the training sequence is the better the match between the predicted and true posterior probabilities will be.

![Fig. 5. Calibration curve for GPC for a $snr$ of 2dB and the channel model in (43) for 100 training samples.](image-url)
Fig. 6. Calibration curve for GPC for a snr of 2dB and the channel model in (43) for 200 training samples.

Fig. 7. Calibration curve for GPC for a snr of 2dB and the channel model in (43) for 800 training samples.

8. The GPR Multi-user detector

The GPR mean prediction in (26) can be directly used as a nonlinear multi-user detector (Murillo-Fuentes & Pérez-Cruz, 2009). This GPR estimate resembles that of a linear detector. It has a weight vector, either \( \mu_w \) or \( C^{-1}s \), multiplied by a nonlinear transformation, either \( \phi^\top(x_s) \) or \( k^\top \), of the input to be predicted. The output of the GPR detector is the prediction
of the bit transmitted by the \textit{UoI}. The input to the GPR is a vector with $n_Q$ consecutives samples from the channel:

$$y_j = \left[ \left( x_j^{mud} \right) \top, \left( x_{j-1}^{mud} \right) \top, \ldots, \left( x_{j-n_Q+1}^{mud} \right) \top \right] \top,$$

where $x_j$ are the $n_S$ chips received at time step $j$ in (7). If the codes of the other users are available, we could first project the received chips onto them as follows,

$$y_j = \left[ \left( x_j^{mud} \right) \top H_S, \left( x_{j-1}^{mud} \right) \top H_S, \ldots, \left( x_{j-n_Q+1}^{mud} \right) \top H_S \right] \top.$$

The GPR equalizer is trained with a set $D = \{y_i, s_i\}_{i=1}^n$. For a new input $y_*$, the \textit{UoI} symbol is estimated as

$$\hat{s}_s(l) = \text{sign}(\mu_{s_l}) = \text{sign}(k^\top C^{-1} s),$$

where

$$k = [k(y_*, y_1), \ldots, k(y_*, y_n)] \top.$$

As mentioned earlier, the Gaussian processes framework can be extended for solving classification tasks. Gaussian process for classification (GPC) solution is non-analytical and we need to approximate its posterior distribution to make posterior probability predictions for the new samples and to train its hyperparameters. These approximations are computationally intensive and make GPC harder to train than GPR. Moreover, as detailed in (Williams & Rasmussen, 2006) and illustrated in Fig. 4 in the last section, in many cases GPR performs equally well to GPC, if we are only interested in minimizing the misclassification rate. Therefore, although GPC seems the natural tool for solving this task, we decided to use GPR because it is less computationally demanding and its misclassification rate is similar to that of GPC.

### 8.1 Experimental results

In our first experiment, we employ Gold spreading codes with 31 chips per user, because they have favorable cross correlation properties that limit the interferences by other users and their delayed replicas (Cover & Thomas, 1991). We report results for systems operating with 3 and 16 users and we assume the user of interest is 50 dB below the other users. This is a fairly standard scenario when one of the users is close to the base station and it is assigned little power. We use the received 31 chips to detect each transmitted symbol. The channel model is the one described in (41) and (42) in Section 0.
We show the bit error rate (BER) versus the $snr$ for 16 users in Fig. 8 with 512 training symbols. The effect of the 16 users can be approximated by white noise. Therefore, the optimal solution will be closed to a linear detector. All the receivers perform similarly well except for the nonlinear SVM. The training sequence for the nonlinear SVM with 16 users is not long enough, and hence the nonlinear SVM is unable to detect the transmitted bits and reports chance-level performances. The GPR solution is quite similar to the MMSE solution, because it almost shuts down its nonlinear part in (38). As we show in Section 0, the GPR with a linear kernel and the linear MMSE provide equivalent solutions in this case. This result is quite relevant, as we do not tell the GPR receiver that the solution is linear. GPR finds it out on its own, when it maximizes the hyperparameters’ likelihood. The GPC also cancels its nonlinear part and it is able to avoid overfitting. The linear SVM detector presents the worse performance among the proposed methods that converge in both cases, although it is barely noticeable in the figures.

Fig. 8. We report the BER versus the $snr$ for a multiuser detector with 16 users. The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash-dotted line the GPC, the dotted line with circles the linear SVM, and the dotted line with bullets the nonlinear SVM.

The optimal solution is almost linear and all the proposed procedures perform equally well, once the training sequence is long enough. The training sequence of 512 symbols is not long enough for the nonlinear SVM with 16 users and it is unable to correctly tune its multiuser detector. If we had increased the training sequence to several thousand samples, the nonlinear SVM would converge and it would provide a solution close to the other algorithms. The differences in BER are not significant to decide which method is best, but the differences in training time might lead us to choose one over the others, as we discuss in short.

We report the BER as a function of the training examples for 16 users and $snr = 16\, dB$ in Fig. 9. These results are more meaningful than the BER versus $snr$ reported in Fig. 8, because there is a significant disparity between the performances of the different methods.
The GPR receiver presents the fastest learning curve closely followed by the linear MMSE and linear SVM solutions. We conjecture this is due to the GPR optimal training of its hyperparameter, because it is able to adjust them for each training sequence, while the linear SVM uses a constant setting, which might be good for a long training sequence, but not as good for shorter ones.

Fig. 9. We report the BER versus the length of the training sequence for a multiuser detector with 16 users and $\text{snr} = 16$ dB. The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash-dotted line the GPC, the dotted line with circles the linear SVM, and the dotted line with bullets the nonlinear SVM.

We repeat Experiment 2 in (Chen et al, 2001), in which 3 users transmit with an orthogonal 8-dimension spreading code. The solution for user 2 is highly nonlinear and we report the BER versus the $\text{snr}$ in Fig. 10. The linear SVM and MMSE clearly underperform compared to the nonlinear methods. The GPR and nonlinear SVM achieve almost identical results. The GPC for low $\text{snr}$ mimics the results of the nonlinear methods ($\text{snr} < 14$ dB) and for high $\text{snr}$, it reports the same results as the linear receivers ($\text{snr} > 16$ dB). This behavior is explained by the length and diversity of the training sequence. If the training sequence is long enough, the GPC receiver provides the best nonlinear decision function, otherwise it reports the best linear decision function to avoid overfitting. For low $\text{snr}$, 512 symbols are long enough for the GPC to achieve the best nonlinear decision function and the GPC receiver trains its hyperparameters to obtain this nonlinear detector. For high $\text{snr}$, there is not enough diversity in a training sequence with 512 symbols and it is only able to report the best linear detector, as it shuts down its nonlinear part to avoid overfitting. With these two experiments, we are able to show that the GPR with the covariance function in (38) is able to obtain the best results in both scenarios. If the solution is linear, it performs as the linear MMSE, needing shorter-training sequences. If the solution is nonlinear, the GPR receiver builds a nonlinear detector that significantly improves the MMSE solution.

10. Acknowledgements

This work was partially funded by Spanish government (Ministerio de Educación y Ciencia TEC2006-13514-C02-{01,02}/TCM, Consolider-Ingenio 2010 CSD2008-00010) and the
The GPR receiver presents the fastest learning curve closely followed by the linear MMSE and linear SVM solutions. We conjecture this is due to the GPR optimal training of its hyperparameter, because it is able to adjust them for each training sequence, while the linear SVM uses a constant setting, which might be good for a long training sequence, but not as good for shorter ones.

Fig. 9. We report the BER versus the length of the training sequence for a multiuser detector with 16 users and $16\text{snr}=\text{dB}$. The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash-dotted line the GPC, the dotted line with circles the linear SVM, and the dotted line with bullets the nonlinear SVM.

We repeat Experiment 2 in (Chen et al, 2001), in which 3 users transmit with an orthogonal 8-dimension spreading code. The solution for user 2 is highly nonlinear and we report the BER versus the $\text{snr}$ in Fig. 10. The linear SVM and MMSE clearly underperform compared to the nonlinear methods. The GPR and nonlinear SVM achieve almost identical results. The GPC for low $\text{snr}$ mimics the results of the nonlinear methods ($14\text{snr}<\text{dB}$) and for high $\text{snr}$, it reports the same results as the linear receivers ($16\text{snr}>\text{dB}$). This behavior is explained by the length and diversity of the training sequence. If the training sequence is long enough, the GPC receiver provides the best nonlinear decision function, otherwise it reports the best linear decision function to avoid overfitting. For low $\text{snr}$, 512 symbols are long enough for the GPC to achieve the best nonlinear decision function and the GPC receiver trains its hyperparameters to obtain this nonlinear detector. For high $\text{snr}$, there is not enough diversity in a training sequence with 512 symbols and it is only able to report the best linear detector, as it shuts down its nonlinear part to avoid overfitting.

With these two experiments, we are able to show that the GPR with the covariance function in (38) is able to obtain the best results in both scenarios. If the solution is linear, it performs as the linear MMSE, needing shorter-training sequences. If the solution is nonlinear, the GPR receiver builds a nonlinear detector that significantly improves the MMSE solution.

Fig. 10. We report the BER versus $\text{snr}$ for a multiuser detector with 3 users and a training sequence of 512 symbols. The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash-dotted line the GPC, the dotted line with circles the linear SVM and the dotted line with bullets the nonlinear SVM. The linear SVM is on top of the linear MMSE line.

9. Conclusions

We have proposed GPR and GPC for designing digital communication receivers. GPR follows a wide range of machine learning tools that have been successfully applied to the design of digital communication receivers. GPR can be viewed as a nonlinear MMSE. MMSE is the standard criterion used for designing digital communication receivers, as it trades off inverting the channel and not amplifying the noise. GPR solution is analytical given the nonlinear function, while most machine-learning methods need to perform an optimization problem to achieve their solution. On the other hand, GPC provides extra information for each one of its decisions, i.e. the posterior probability of being in the correct class. This information can be used by the channel decoder to significantly reduce the BER for low signal to noise ratio. This characteristic is not shared by the other nonlinear machine learning tools, as they can only provide hard decisions as outputs. We have shown that, as the number of samples increases, the predicted probabilities tend to the true posterior probabilities. To highlight the advantages of GPs as digital communications receivers we compare their performances to that of SVM. SVM provides solutions as good as GPR does, but it needs more training samples. These tools have been compared in two typical scenarios in digital communications: equalization and multiuser detection. In both experiments GPs exhibit an outstanding behavior.

10. Acknowledgements

This work was partially funded by Spanish government (Ministerio de Educación y Ciencia TEC2006-13514-C02-{01,02}/TCM, Consolider-Ingenio 2010 CSD2008-00010) and the
European Union (FEDER). Fernando Pérez-Cruz is supported by Marie Curie Fellowship 040883-AI-COM

11. References


The goal of this book is to present the latest applications of machine learning, which mainly include: speech recognition, traffic and fault classification, surface quality prediction in laser machining, network security and bioinformatics, enterprise credit risk evaluation, and so on. This book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides them with a good introduction to many application researches of machine learning, and it is also the source of useful bibliographical information.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following: