Digital Communication Receivers Using Gaussian Processes for Machine Learning

Fernando Pérez-Cruz and Juan José Murillo-Fuentes [*]

May 27, 2008

Abstract

We propose Gaussian processes (GPs) as a novel nonlinear receiver for digital communication systems. The GPs framework can be used to solve both classification (GPC) and regression (GPR) problems. The optimal minimum mean squared error solution is the expectation of the transmitted symbol given the information at the receiver, which is a nonlinear function of the received symbols for discrete inputs. GPR can be presented as a nonlinear MMSE estimator and thus capable of achieving optimal performance from MMSE viewpoint. Also, the design of digital communication receivers can be viewed as a detection problem, in which GPC is specially suited as it assigns posterior probabilities to each possible transmitted symbol. In this paper, we explore the suitability of GPC and GPR as nonlinear digital communication receivers. The major advantage of GPs is that they are Bayesian

[*]This work was partially funded by the Spanish government (Ministerio de Educación y Ciencia TEC2006-13514-C02-01/TCM and TEC2006-13514-C02-02/TCM), the European Union (FEDER) and the Comunidad de Madrid (project 'PRO-MULTIDIS-CM', id. S0505/TIC/0223). Fernando Pérez-Cruz is supported by Marie Curie Fellowship 040883-AI-COM. F. Pérez-Cruz is in the Electrical Engineering Department at Princeton University. Princeton, NJ 08544. He is also an Associate Professor in Universidad Carlos III de Madrid. E-mail: `fp@princeton.edu` J.J. Murillo-Fuentes is with the Dept. Teoría de la Señal y Comunicaciones, Escuela Técnica Superior de Ingenieros, Universidad de Sevilla, Paseo de los Descubrimientos s/n, 41092 Sevilla, Spain. E-mail: `murillo@us.es`

machine learning tools that allow a formulation of a likelihood function for its hyperparameters, which can then be set optimally. Thereby, it uses an optimal hyperparameter setting to achieve the best nonlinear receiver for each training sequence. GPs outperform state-of-the-art nonlinear machine learning approaches that prespecify their hyperparameters or rely on cross-validation, whose computational complexity is high and unpredictable. We illustrate the advantages of GPs as digital communication receivers for linear and nonlinear channel models for short training sequences and compare them to other state-of-the-art nonlinear machine learning tools such as support vector machines.

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 [28]. In this paper, 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 [22] and were rapidly forgotten due to its computation complexity. In the mid-nineties, they were independently rediscovered [39]. Since then they have been shown to fit many different applications [31] and nowadays their computational complexity is no longer a limiting issue [29].

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: multi-layered perceptrons (MLPs) [10], radial basis function networks (RBFNs) [5], recurrent RBFNs [7], self-organizing feature maps (SOFMs) [16], wavelet neural networks [4], GCMAC [13], kernel adaline (KA) [20] or support vector machines (SVMs) [27], among many others. Other digital communication systems that have also benefited from nonlinear detection and estimation algorithms are multi-user detection [9, 35], multiple-input multiple-output systems [32], beam forming [19], pre-distortion [12] and plant identification [1], 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 [15, 2]) 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. Also, it makes the length of the training sequence hard to predict, as it depends on how well the chosen structure or hypeparameters fits the current problem.

For example, SVM with a Gaussian kernel needs to fit its width, which is proportional to the noise level [20, 27, 6]. If the width is too large, the SVM can be optimized with short training sequences, but its performance is poor. If it is too small, it requires a significantly longer training sequence to avoid overfitting. For each instantiation of the problem there is an optimal width. This kernel width depends not only on the channel values and noise level, as we would expect, but also on the actual values of the noise themselves. Ideally, we would like to choose the kernel width every time we receive a new training sequence. But this would involve training a different SVM for each possible width and then choosing the optimal receiver (validation). In addition, this width is not the only SVM's hyperparameter. We must also validate the soft-margin that trades off the minimization of the training errors and the maximization of the margin. Therefore, we would have to train a set of receivers with different width and soft-margin hyperparameters to find the optimal setting in each problem. However, typically, we can only solve a single optimization problem in the receiver. We thus prespecify the SVM hyperparameters, as it is the case with other nonlinear tools referenced earlier.

In previous work, we introduced Gaussian processes for machine learning as a novel nonlinear tool for designing digital communication receivers. Gaussian processes can be applied to regression and classification problems [31] and in this paper we use both settings for tuning digital communication receivers with short training sequences. We compare Gaussian processes for regression (GPR) and Gaussian processes for classification (GPC) to state-of-the-art linear and nonlinear receivers to show their strength in solving this relevant problem. We have presented some preliminaries results for multi-user detection in CDMA systems [21, 26] and channel equalization in [3]. In this paper we extend these results and include GPC in our comparisons.

Gaussian processes for machine learning are rooted in Bayesian statistics [31] and, consequently, build 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. We focus on short training sequences and show that fixed hyperparameters underperform compared to GPR receivers with optimally trained hyperparameters.

Gaussian processes can be extended for solving classification problems. In this case the posterior is no longer tractable and we need to use approximations to compute the prediction for each class label [31]. A Gaussian distribution is typically used to approximate the GPC's posterior, either using Laplace [38] or expectation propagation methods [17]. However, GPC computational complexity is significantly higher than that of GPR and hence they might not be as suited for designing digital communication receivers as GPR are. Moreover, their performance is not as good as that of GPR receivers as we show and explain in the experimental section.

The rest of the paper is organized as follows. We present the design of digital communication receivers as an optimization problem in Section 2 and show how different nonlinear machine learning tools can be fitted in this framework. Section 3 is devoted to Gaussian processes for regression and how it can be understood as a nonlinear MMSE estimation. The optimization of the GPR hyperparameters is proposed in Section 4. GPC are introduced briefly in Section 5. We present some computer simulations in Section 6 to illustrate the benefits of GPR for channel equalization and multi-user detection compared to other state-of-the-art nonlinear tools. We conclude with some final remarks and proposed further work in Section 7.

2 Nonlinear optimization for communication receivers

2.1 Channel model and MMSE

We consider throughout the paper the following deterministic channel model:

$$\mathbf{x} = \mathbf{H}\mathbf{s} + \mathbf{z} \tag{1}$$

where $\mathbf{s}$ is a random variable column-vector representing the transmitted symbols, $\mathbf{H}$ corresponds to the deterministic channel gains, unknown to both the transmitter and receiver, $\mathbf{z}$ is zero-mean Gaussian noise, and $\mathbf{x}$ represents the received symbols. This model is general enough to capture most standard communication systems. For example:

- *Inter Symbol Interference*: Each element in $\mathbf{s}$ is a symbol transmitted at a different time instant. $\mathbf{H}$ is a Toeplitz matrix, in which each row represents the channel impulsive response.

- *Multiple-Input Multiple-Output*: $(\mathbf{H})_{ij}$ represents the gain from the i^{th} receiving antenna to the j^{th} transmitting antenna and $\mathbf{s}$ represents the symbols transmitted by the antenna array.

- *Fading*: $\mathbf{H}$ is a diagonal matrix with the fading coefficients and $\mathbf{s}$ represents the symbols transmitted at each time instant.

- *CDMA*: The columns of $\mathbf{H}$ collect each user's spreading code and each element of $\mathbf{s}$ represents the symbol transmitted by the users.

We can also combine different $\mathbf{H}$ matrices to accommodate other communication systems. For example $\mathbf{H} = \mathbf{H}_1\mathbf{H}_2\mathbf{H}_3$, where $\mathbf{H}_1$ is a Toeplitz matrix representing an inter-symbol interference channel model, $\mathbf{H}_2$ contains the spreading codes of a CDMA system, and $\mathbf{H}_3$ is a diagonal matrix assigning different power to each user. This $\mathbf{H}$ matrix represents the downlink channel in a mobile communication network.

The source $\mathbf{s}$ that achieves capacity (maximum information transmission rate) [8] is a zero-mean Gaussian distribution with a covariance matrix given by the right eigenvectors of the channel matrix [30]. $\mathbf{s}$ being a continuous random variable, we can estimate in the receiver the transmitted vector using a minimum mean squared error (MMSE) detector:

$$f_{mmse}(\mathbf{x}) = \underset{f(\cdot)}{\operatorname{argmin}}\, E\left[\|\mathbf{s} - f(\mathbf{x})\|^2\right] \tag{2}$$

The function $f_{mmse}(\mathbf{x})$ is the mean value of $\mathbf{s}$ given the received vector $\mathbf{x}$, $E[\mathbf{s}|\mathbf{x}]$, which is a linear function of $\mathbf{x}$ if $\mathbf{s}$ is Gaussianly distributed. 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 $E[\mathbf{s}|\mathbf{x}]$ if $\mathbf{s}$ is a discrete random variable[1], 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 solution is given by:

$$\mathbf{w}_{mmse} = \underset{\mathbf{w}}{\operatorname{argmin}}\, E\left[\left(s - \mathbf{w}^\top\mathbf{x}\right)^2\right] = \left(E\left[\mathbf{x}\mathbf{x}^\top\right]\right)^{-1} E\left[\mathbf{x}s\right]. \tag{3}$$

If $\mathbf{H}$ is unknown, we can replace the expectations by sample averages using a training sequence.

[1]Even for a simple example, if $s \in \{\pm 1\}$ and equally likely and $\mathbf{H} = 1$ then $E[s|x] = \tanh(x/\sigma_z^2)$.

2.2 Machine learning for digital communication receivers

The design of digital communication receivers can be readily understood as a supervised classification problem [10, 23], in which the receiver constructs a classifier for deciding over the incoming symbols. Machine learning tools optimize the risk of misclassification:

$$f_{opt}(\mathbf{x}) = \underset{f(\cdot)}{\operatorname{argmin}}\, E\left[L\left(s, f(\mathbf{x})\right)\right] = \underset{f(\cdot)}{\operatorname{argmin}} \int L\left(s, f(\mathbf{x})\right) p(s, \mathbf{x}) ds d\mathbf{x}, \qquad (4)$$

where $L(\cdot)$ is a loss-function that measures the penalty for wrongly classifying a pattern and $f(\mathbf{x})$ is the nonlinear model to predict s.

The joint density, $p(s, \mathbf{x})$, is typically unknown and thus we use a training sequence $\{\mathbf{x}_i, s_i\}_{i=1}^{n}$ and the empirical risk minimization (ERM) inductive principle [36] to obtain the optimal solution:

$$\widehat{f}_{opt}(\mathbf{x}) = \underset{f(\cdot)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(s_i, f(x_i)) + \lambda \Omega(||f||) \right\}, \qquad (5)$$

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 [36] as the number of training samples increases. The number of training patterns n determines the symbols in the preamble of each transmission needed to adjust the receiver. This number should be small to maximize the number of bits used to transmit information, as we need to retransmit the preamble in each burst of data.

The nonlinear machine learning approaches mentioned in the introduction can be cast as the optimization in (5) using an appropriate nonlinear model, loss-function and regularizer. For example: $f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is a nonlinear transformation to a higher dimensional space; $L(s_i, f(x_i)) = (1 - s_i \mathbf{w}^\top \mathbf{x}_i)_+$, hinge-loss[2]; and $\Omega(||f||)) = ||\mathbf{w}||^2$ weight decay [2], gives an SVM for a binary antipodal constellation, which constructs the nonlinear classifier using the 'kernel trick' for $\boldsymbol{\phi}(\cdot)$ [34].

The convexity of the optimization in (5) depends on $f(\cdot)$, $L(\cdot,\cdot)$ and $\Omega(\cdot)$. In some cases, as in SVM or KA, it leads to a convex functional and in others, as in

[2] $(y)_+ = \max(y, 0)$

MLP or RBFN, it does not. But in any case, these machine learning approaches rely on an iterative optimization tool [2, 34] for solving (5).

If we choose $f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$, $L(s, f(\mathbf{x})) = (s - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}))^2$ and $\Omega(f) = ||\mathbf{w}||^2$ we get a convex functional:

$$\mathbf{w}_{nl\_mmse} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (s_i - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i))^2 + \lambda ||\mathbf{w}||^2 \right\} \tag{6}$$

that can be analytically optimized:

$$\mathbf{w}_{nl\_mmse} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I} \right)^{-1} \boldsymbol{\Phi}^\top \mathbf{s}. \tag{7}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_n)]^\top$ and $\mathbf{s} = [s_1, \ldots, s_n]^\top$. We denote this solution as nonlinear MMSE, since it is a nonlinear extension of (3), in which we have substituted $\mathbf{x}$ by $\boldsymbol{\phi}(\mathbf{x})$ and we have replaced the expectations by sample averages.

In the next section we show (7) 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 [24]. Moreover, interpreting (7) as GPR allows optimizing its hyperparameters by maximum likelihood (Section 4). This optimization improves the performance of (7) with respect to other nonlinear machine learning procedures when the number of training samples is low, because for reduced training datasets the performance of nonlinear machine learning methods significantly depends on its hyperparameters.

3 Gaussian Processes for Regression

In the past few years a new Bayesian machine learning tool based on Gaussian processes (GPs) has been developed for nonlinear regression estimation [39, 31, 37]. 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 present GPR from the Bayesian generalized linear regression viewpoint. Although from this opening we lose the GPs interpretation and we can only work with Gaussian likelihood models, we believe it is a simpler way to understand GPR. This approach mimics how most machine learning textbooks introduce nonlinear regression [2, 34, 14] 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. Both interpretations are described in [37], where they are shown to be identical for Gaussian likelihood models. There is more to GPs than what we introduce in this summary, for interested readers GPs extensions can be found in [31].

A generalized linear regressor expresses the input-output relation as

$$s = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + \nu, \tag{8}$$

where $\boldsymbol{\phi}(\cdot)$ is a nonlinear transformation to a higher dimensional feature space and ν is a random variable that measures the deviation between s and its estimate. 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 ν, we can compute the regressor $\mathbf{w}$ by maximum likelihood (ML),

$$\mathbf{w}_{ML} = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^n p(\nu_i) = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^n p(s_i - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i)). \tag{9}$$

We use these ML weights to predict the outputs for future test points x_*:

$$s_* = \mathbf{w}_{ML}^\top \boldsymbol{\phi}(\mathbf{x}_*). \tag{10}$$

In Bayesian machine learning $\mathbf{w}$ is considered to be a random variable and, to predict the outcome of $\mathbf{x}_*$, we use its conditional density given the training dataset, $p(\mathbf{w}|\mathcal{D})$. This conditional density, known as the posterior of $\mathbf{w}$, can be computed through Bayes rule,

$$p(\mathbf{w}|\mathcal{D}) = p(\mathbf{w}|\mathbf{s}, \mathbf{X}) = \frac{p(\mathbf{s}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{s}|\mathbf{X})} = \frac{p(\mathbf{w})}{p(\mathbf{s}|\mathbf{X})} \prod_{i=1}^n p(s_i|\mathbf{x}_i, \mathbf{w}), \tag{11}$$

where $p(s_i|\mathbf{x}_i, \mathbf{w})$ is the likelihood function of $\mathbf{w}$, $p(\mathbf{w})$ its prior distribution and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$.

To predict the output for a new test point $\mathbf{x}_*$ we integrate out $\mathbf{w}$:

$$p(s_*|\mathbf{x}_*, \mathcal{D}) = \int_{\mathcal{W}} p(s_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}, \tag{12}$$

in which the conditional density of each s_* (the likelihood of $\mathbf{w}$) is weighted by the posterior of $\mathbf{w}$ and sum 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, $p(s|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \sigma_\nu^2)$, leads to the MMSE criterion, and a zero-mean Gaussian prior, $p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{w}}^2 \mathbf{I})$, allocates probability mass to every possible $\mathbf{w}$ and allows solving (12) analytically. The posterior distribution in (11) is then a Gaussian density function, $p(\mathbf{w}|\mathcal{D}) \sim \mathcal{N}(\mu_{\mathbf{w}}, \boldsymbol{\Sigma}_{\mathbf{w}})$, where

$$\mu_{\mathbf{w}} = \sigma_{\mathbf{w}}^2 \left(\sigma_{\mathbf{w}}^2 \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma_\nu^2 \mathbf{I} \right)^{-1} \boldsymbol{\Phi}^\top \mathbf{s}, \tag{13}$$

$$\boldsymbol{\Sigma}_{\mathbf{w}}^{-1} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} / \sigma_\nu^2 + \mathbf{I} / \sigma_{\mathbf{w}}^2. \tag{14}$$

Actually, the posterior mean in (13) is identical to the maximum a posteriori (MAP) of (11):

$$\begin{aligned} \mu_{\mathbf{w}} &= \mathbf{w}_{MAP} = \underset{\mathbf{w}}{\operatorname{argmax}} \{p(\mathbf{w}|\mathbf{s}, \mathbf{X})\} \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \{\log p(\mathbf{s}|\mathbf{X}, \mathbf{w}) + \log p(\mathbf{w})\} \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \left\{ -\frac{1}{\sigma_\nu^2} \sum_{i=1}^{n} (s_i - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i))^2 - \frac{1}{\sigma_{\mathbf{w}}^2} ||\mathbf{w}||^2 \right\}, \end{aligned} \tag{15}$$

which is identical to (6) for $\lambda = \sigma_\nu^2 / \sigma_{\mathbf{w}}^2$. We can also check that (13) is equal to (7). Therefore the GPR mean prediction can be regarded as a nonlinear MMSE estimation for the nonlinear mapping $\boldsymbol{\phi}(\cdot)$.

The prediction for s_* in (12) is a Gaussian density function, $p(s_*|\mathbf{x}_*, \mathcal{D}) \sim \mathcal{N}(\mu_{s_*}, \sigma_{s_*})$:

$$\mu_{s_*} = \boldsymbol{\phi}^\top(\mathbf{x}_*)\mu_{\mathbf{w}} = \boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\Phi}^\top\mathbf{s}/\sigma_\nu^2 \tag{16}$$

$$\sigma_{s_*}^2 = \boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\phi}(\mathbf{x}_*) = \boldsymbol{\phi}^\top(\mathbf{x}_*)\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}/\sigma_\nu^2 + \mathbf{I}/\sigma_{\mathbf{w}}^2\right)^{-1}\boldsymbol{\phi}(\mathbf{x}_*). \tag{17}$$

There is an alternative formulation for μ_{s_*} and $\sigma_{s_*}^2$, in which we do not need to know the nonlinear mapping $\boldsymbol{\phi}(\cdot)$ and we only need to work with its inner product or kernel, defined as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\mathbf{w}}^2\boldsymbol{\phi}^\top(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j). \tag{18}$$

To obtain this alternative formulation, we first define the covariance matrix $\mathbf{C}$ as:

$$(\mathbf{C})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_\nu^2\delta_{ij}, \tag{19}$$

which can be related to $\Sigma_{\mathbf{w}}$ as follows:

$$\boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\boldsymbol{\Phi}^\top = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}/\sigma_\nu^2 + \mathbf{I}/\sigma_{\mathbf{w}}^2\right)\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}^\top\left(\sigma_{\mathbf{w}}^2\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma_\nu^2\mathbf{I}\right)/(\sigma_\nu^2\sigma_{\mathbf{w}}^2) = \boldsymbol{\Phi}^\top\mathbf{C}/(\sigma_\nu^2\sigma_{\mathbf{w}}^2). \tag{20}$$

Now if we pre-multiply (20) by $\boldsymbol{\Sigma}_{\mathbf{w}}$ and post-multiply it by $\mathbf{C}^{-1}$, we obtain the following equivalency: $\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\Phi}^\top/\sigma_\nu^2 = \sigma_{\mathbf{w}}^2\boldsymbol{\Phi}^\top\mathbf{C}^{-1}$, which can be used to simplify (16) and express the GPR prediction mean as:

$$\mu_{s_*} = \boldsymbol{\phi}^\top(\mathbf{x}_*)\sigma_{\mathbf{w}}^2\boldsymbol{\Phi}^\top\mathbf{C}^{-1}\mathbf{s} = \mathbf{k}^\top\mathbf{C}^{-1}\mathbf{s}, \tag{21}$$

where

$$\mathbf{k} = \sigma_{\mathbf{w}}^2\boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{\Phi}^\top = [k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_n)]^\top. \tag{22}$$

To compute the prediction for any vector $\mathbf{x}_*$, we do not need to know the nonlinear mapping $\boldsymbol{\phi}(\cdot)$, only its kernel. The complexity of computing μ_{s_*} in (21) is linear, because we can pre-compute the vector $\mathbf{C}^{-1}\mathbf{s}$ that does not depend on $\mathbf{x}_*$ and we only need to filter $\mathbf{k}$ with it for each new test pattern.

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

$$\sigma_{s_*}^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^\top \mathbf{C}^{-1}\mathbf{k}, \tag{23}$$

which is achieved after applying the matrix inversion lemma [33] to (14).

Equations in (21) and (23) represent the predictions for $\mathbf{x}_*$ given by the Gaussian processes view of GPR. The matrix $\mathbf{C}$ is the covariance matrix of a multidimensional Gaussian distribution, hence its name, that describes the training data, and the vector $\mathbf{k}$ represents the covariance vector between the training dataset and the test vector. Therefore, the function $k(\cdot,\cdot)$ has to be a positive-definite function to ensure that the Gaussian processes covariance matrix $\mathbf{C}$ is also positive-definite.

4 Hyperparameter optimization

If either $\boldsymbol{\phi}(\cdot)$ or $k(\cdot,\cdot)$ are known, we can analytically predict the output of any incoming sample using (21). But for most estimation problems the best nonlinear transformation (or its kernel) is unknown. As discussed in the Section 2, the optimal setting of the hyperparameters could be obtained by cross-validation, similarly to any other nonlinear machine learning method. In this case the nonlinear MMSE would be as good as any of the other methods, as it would require either to try different settings or to rely on a prespecify one.

From the point of view of Bayesian machine learning, we can proceed as we did for the parameters $\mathbf{w}$ in Section 3. First, we compute the likelihood of the hyperparameters of the kernel given the training dataset:

$$p(\mathbf{s}|\mathbf{X},\theta) = \int p(\mathbf{s}|\mathbf{w}\mathbf{X},\theta)p(\mathbf{w}|\mathcal{D},\theta)d\mathbf{w} = \frac{1}{\sqrt{(2\pi)^n|\mathbf{C}_\theta|}}\exp\left(-\frac{1}{2}\mathbf{s}^\top\mathbf{C}_\theta^{-1}\mathbf{s}\right), \tag{24}$$

where θ represents the hyperparameters of the covariance function or kernel. We have added θ to the covariance matrix, likelihood and posterior to explicitly indicate that they depend on the kernel's hyperparameters. This was omitted in the GPR presentation in Section 3 for clarity purposes.

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

$$p(\theta|\mathcal{D}) = \frac{p(\mathbf{s}|\mathbf{X},\theta)p(\theta)}{p(\mathbf{s}|\mathbf{X})}. \tag{25}$$

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

$$p(s_*|\mathbf{x}_*,\mathcal{D}) = \int p(s_*|\mathbf{x}_*,\mathcal{D}\theta)p(\theta|\mathcal{D})d\theta. \tag{26}$$

However, in this case, the hyperparameters' likelihood 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 computational intensive and it is not feasible for digital communications receivers. For example, Markov-chain Monte Carlo (MCMC) methods require several hundreds to several thousands samples from the posterior of θ to integrate it out in (26). For the interested readers, further details can be found in [31].

Alternatively, we can use the likelihood function of the hyperparameters and compute its maximum to obtain its optimal setting [39], 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:

$$\theta_{ML} = \operatorname*{argmax}_{\theta}\ p(\mathbf{s}|\mathbf{X},\theta) = \operatorname*{argmax}_{\theta}\ \log p(\mathbf{s}|\mathbf{X},\theta) = \operatorname*{argmax}_{\theta}\left\{-\mathbf{s}^\top\mathbf{C}_\theta^{-1}\mathbf{s} - \log|\mathbf{C}_\theta|\right\} \tag{27}$$

This optimization is non-convex [18]. 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 [31] for further details.

4.1 Covariance matrix

To optimize the kernel hyperparameters in (27) 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 [31], a list of flexible kernels –i.e. linear, Gaussian, neural networks, Matérn, among others– and their properties are described. The rules on how to combine them are also described –e.g. the sum or product of two kernel functions is also a valid kernel function–.

For example, if we know the optimal solution to be linear, we could use the linear kernel: $k(\mathbf{x}, \mathbf{x}') = \sigma^2_{\mathbf{w}} \mathbf{x}^\top \mathbf{x}'$. The only unknown hyperparameters in this case are σ^2_ν and $\sigma^2_{\mathbf{w}}$, as we do not need to know these variances a priori. In the remaining of this text, we consider, without loss of generality, the last term in (19) to be part of the designed kernel, as δ_{ij} is a valid kernel and the weighted sum of kernel functions (with nonnegative weights) is also a kernel. 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, γ_ℓ, per input dimension:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha_1 \exp\left(-\sum_{\ell=1}^{d} \gamma_\ell \left(x_{i\ell} - x_{j\ell} \right)^2 \right) + \alpha_2 \mathbf{x}_i^\top \mathbf{x}_j + \alpha_0 \delta_{ij}. \tag{28}$$

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 $\theta = [\log \alpha_0, \log \alpha_1, \log \alpha_2, \log \gamma_\ell]$, 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 if we work over θ.

The covariance function in (28) is a good kernel for designing digital communication receivers using GPR, because it contains a linear and a universal nonlinear part, as the RBF kernel has an infinite VC dimension [36]. The proposed covariance function is a good match for designing digital communication receivers. 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 $\mathbf{x}$. If the channel is linear, then the ML solution sets $\alpha_1 = 0$ and there is no interference of the nonlinear term with the nonlinear one in the solution. Also, using a radial basis 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.

4.2 Discussion

Gaussian Processes for regression is a nonlinear regression tool that, given the covariance function, provides an analytical solution to any regression estimation problem. Moreover, it does not only give point estimates, but it also assigns confidence intervals for them. In GPR, we perform the optimization step to set the covariance function hyperparameters by maximum likelihood, unlike SVM or other nonlinear machine learning tools, in which the optimization is used to set the optimal parameters. In these methods, the hyperparameters have to be either prespecified or estimated by cross-validation [15].

Cross-validation optimizes several functionals (typically less than 10) for each possible setting of the hyperparameters [2]. The number of hyperparameters that can be tuned is quite limited (at most 2 or 3), as the computational complexity of cross-validation increases exponentially with the number of hyperparameters. These remarkable drawbacks limit the application of these nonlinear tools to digital communications receivers, since we face complex nonlinear problems with reduced computational resources and short training sequences. By exploiting the GPs framework, as stated in this paper, we can avoid them.

5 Gaussian process 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 [31]. 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 for an input $\mathbf{x}$ the class label as follow:

$$p(s = +1|\mathbf{w}, \mathbf{x}) = p(s = +1|f) = \sigma(f), \tag{29}$$

where $f = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$ is an underlying continuous function, $\sigma(\cdot)$ is a sigmoid[3] that squashes f between 0 and 1, and $p(s = -1|f) = 1 - p(s = +1|f)$.

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 $s_i \in \{\pm 1\}$), we can compute the posterior over the underlying function $\mathbf{f} = [f_1, \ldots, f_n]^\top$ using Bayes rule, as we did in Section 3 for GPR with $\mathbf{w}$, and we can integrate out $\mathbf{f}$ to predict the class label for any new test point $\mathbf{x}_*$. We can compute the class label for the test samples as follows:

$$p(s_* = +1|\mathbf{x}_*, \mathcal{D}) = \int \sigma(f_*) p(f_*|\mathbf{x}_*, \mathcal{D}) df_* \tag{30}$$

where

$$p(f_*|\mathbf{x}_*, \mathcal{D}) = \int p(f_*|x_*, \mathbf{X}, \mathbf{f}) p(\mathbf{f}|\mathcal{D}) d\mathbf{f} \tag{31}$$

and

$$p(\mathbf{f}|\mathcal{D}) = p(\mathbf{f}|\mathbf{X}, \mathbf{s}) = \frac{\prod_i p(\mathbf{s}_i|f_i) p(\mathbf{f}|\mathbf{X})}{p(\mathbf{s}|\mathbf{X})}. \tag{32}$$

In (31) we compute the distribution for the underlying function in the test point and in (30) we integrate out the underlying function to predict the probability that the class label of that point is $+1$. Both integrals are intractable due to the likelihood model employed for f in (29). GPC typically relies on a Gaussian approximation[4] for the posterior density $p(\mathbf{f}|\mathcal{D})$, to analytically solve (31), and (30) is a one-dimensional integral that can be easily solved numerically. Further details on how to approximate the posterior and train the covariance function hyperparameters can be found in [31].

[3]This function is typically the logistic or the cumulative density function of a Gaussian [31].

[4]The standard approximations to the posterior are Laplace or expectation propagation, as explained in [17].

6 Experimental Results

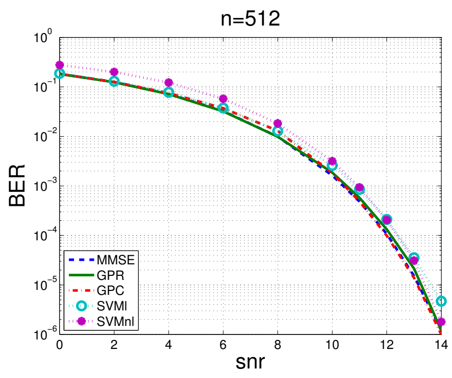
We carry out two sets of experiments. First, we design a receiver for a CDMA system with strong near-far requirements and inter-symbol interference. In the second experiment, we deal with a channel equalization problem with a nonlinear amplifier in the receiver. The results in these experiments allow drawing some general conclusions about the advantages of GPs for designing digital communication receivers. For both experiments the channel model is given by:

$$h(z) = 0.3763 + 0.8466z^{-1} + 0.3763z^{-2}. \tag{33}$$

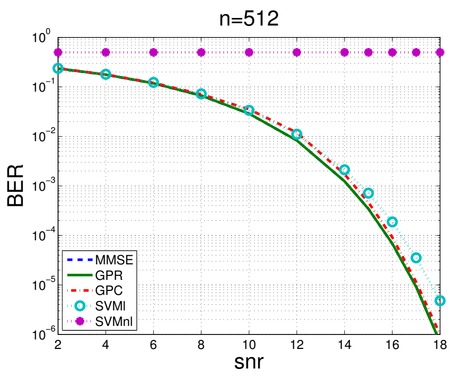
For all these systems we train a linear MMSE receiver (denoted by 'MMSE' and a dashed line), a GPR ('GPR' and a solid line) and a GPC with an EP approximation to its posterior ('GPC' and a dash-dotted line). We approximate the GPC posterior using the EP algorithm, because it provides superior performances than the Laplace approximation as suggested in [17]. For the GPs receivers we work with the covariance matrix in (28). We also report a linear SVM receiver ('SVMl' and a dotted line with circles) and a nonlinear SVM ('SVMnl' and a dotted line with bullets) with an RBF kernel [34]. For the SVMs we train a set of receivers with different hyperparameters and we report the best result. We use $C = 0.5, 1, 2, 5$ and 10 and $\sigma = k\sigma_z$ with $k = 1, 2, 5$ and 10. Thereby, the comparison is biased in favor of the SVM when compared to the GPR and GPC solutions. All the figures are obtained for 100 independently trained trials with 10^5 test symbols.

6.1 Linear multi-user detection

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 [11]. We report results for systems operating with 3 and 16 users and we assume the user of interest is 50dB bellow 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.



(a)

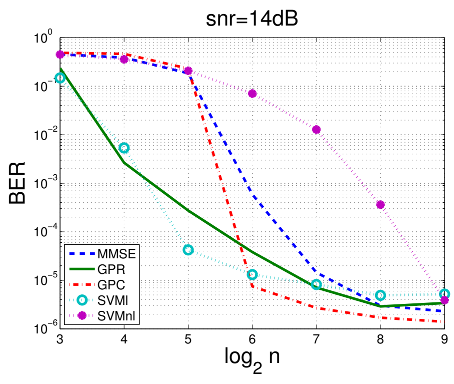


(b)

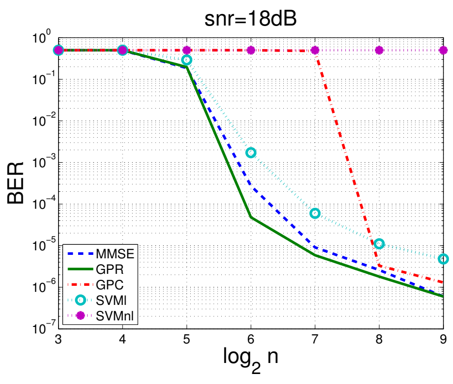
Figure 1: We report the BER versus the snr for a multi-user detector with 3 users in (a) and 16 users in (b). The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash-dotted line the GPC, the dotted line with circles the linear SVM and the dotted line with bullets the nonlinear SVM.

We show the bit error rate (BER) versus the signal to noise ratio (snr) for 3 users in Figure 1(a) and 16 users in Figure 1(b) with 512 training symbols. The solution is almost linear and all the receivers perform similarly well except for the nonlinear SVM for 16 users. 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 (28). As we show in Section 3, 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. It finds 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.

The optimal solution is almost linear and all the proposed procedures perform



(a)



(b)

Figure 2: We report the BER versus the length of the training sequence for a multi-user detector with 3 users and snr=14dB in (a) and 16 users and snr=18dB in (b). 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.

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 multi-user detector. If we had increased the training sequence to several thousands 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 3 users in Figure 2(a) and 16 users in Figure 2(b). For this experiment, these results are more meaningful than the BER versus snr reported in Figure 1, because there is a significant disparity between the performances of the different methods. For 3 users (Figure 2(a)) the GPR and linear SVM are able to reduce the BER for very short training sequences while GPC, MMSE and nonlinear SVM need substantially longer training sequences before they provide non-chance level performances. For 32 training symbols, there are 3 orders of magnitude difference in BER between the former and latter methods.

From these 2 plots, we can easily understand why the nonlinear SVM is unable to converge for 16 users with 512 training symbols. For 3 users the nonlinear SVM needs longer training sequences than the other methods, before it can significantly reduce the BER. For 16 users, the learning problem is harder and it needs several thousands samples to achieve convergence.

The GPR, MMSE and linear SVM learn the solution as the number of training examples increases and they behave almost equally well for 16 users. The GPC needs the training sequence to be long enough before it can produce a meaningful solution. It needs at least 64 symbols for 3 users and 256 for 16 to be able to produce non-chance level performances. But once the training sequence is long enough, it converges to the optimal solution. It does not provide intermediate solutions as the other methods do.

For 16 users, 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.

In this example we can readily understand the advantages of using GPR for solving multi-user detection problems, as for very short training sequences we are able to obtain the best possible solution, and if it is linear, it even improves the linear MMSE solution. The GPR and linear MMSE detectors provide the same solution as the number of samples increases, but for short training sequence the GPR detector is able to optimally set its hyperparameters to provide better performance than the linear MMSE. Also, as we see in the next example, if the solution is nonlinear, it is able to achieve nonlinear multi-user detectors, significantly improving the linear MMSE solution.

6.2 Nonlinear multi-user detection

We repeat the Experiment 2 in [6], 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 snr in Figure 3. 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 snr mimics the results of the nonlinear methods ($snr <$ 14dB) and for high snr it reports the same results as the linear receivers ($snr >$ 16dB). 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 snr, 512 symbols is 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 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. In the first experiment, we already saw that GPC receivers need longer training sequences than GPR, even to achieve the best linear detector. It is clear in this experiment that for nonlinear decision function, GPC receivers even need longer training sequences.

In these two experiments, we are able to show that the GPR with the covariance function in (28) 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 GPC receiver builds a nonlinear detector that significantly improves the linear MMSE and reports the same solution as a nonlinear SVM. The nonlinear SVM is not as good as the GPR with the covariance matrix in (28), because for (almost) linear solutions, it needs significantly longer training sequences, which is a waste of resources in wireless communication systems, as the preamble must be as short as possible. Also a SVM cannot use a kernel as in (28), because it would need to cross validate (or hand pick) too many hyperparameters.

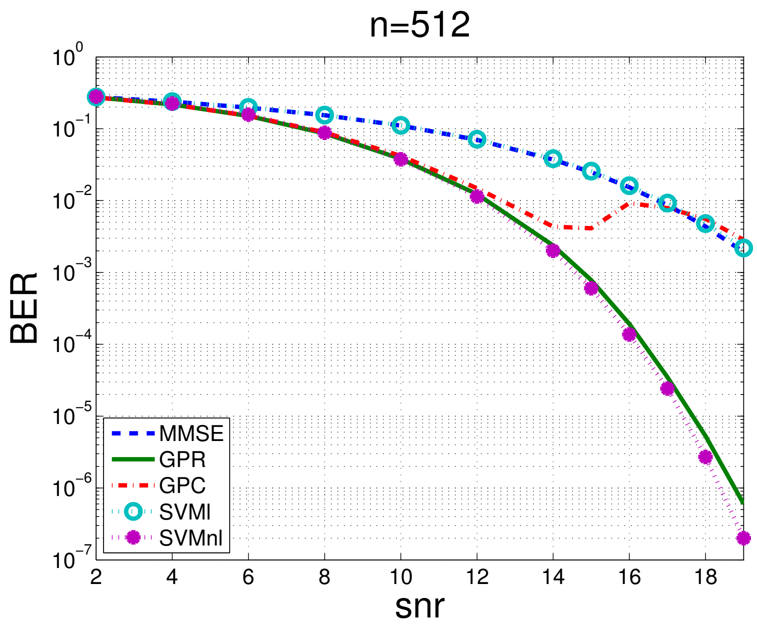


Figure 3: We report the BER versus snr for a multi-user detector with 3 users and a training sequence of 512 symbols. The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash-dotted line the GPC, the dotted line with circles the linear SVM and the dotted line with bullets the nonlinear SVM. The linear SVM is on top of the linear MMSE line.

6.3 Nonlinear channel equalization

Now we turn to the channel equalization problem, in which the channel is represented by (33), and we add a memoryless nonlinearity to the receiver that transforms each received signal as follows:

$$x_i = \hat{x}_i + 0.2\hat{x}_i^2 - 0.1\hat{x}_i^3 + z_i \tag{34}$$

where $\hat{x}_i = (\mathbf{Hs})_i$. This channel model is typically used to described nonlinear amplifiers in wireless communication receivers as explained in [20]. To construct the equalizers, we use 6 received samples to predict each transmitted symbol with a delay of 2 samples.

In Figure 4, we show the BER versus the snr for all equalizers and $n = 512$. For snr less than 22dB the nonlinear GPR equalizer achieves the minimum BER with a gain larger than 3dB for BER around 10^{-3}. For larger snr the performance of this nonlinear equalizer degrades and the linear equalizers perform significantly better. The nonlinear SVM equalizer performs as the GPR equalizer for snr lower than 17dB, but for larger snr the training sequence is not long enough and its solution degrades (overfiting). For snr larger than 20dB, the nonlinear SVM equalizer is not able to reduce the achieved BER. The nonlinear SVM and the GPR as the snr 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 snr, 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 snr. If we had increased the training sequence, the SVM and GPR would perform better than the linear methods for larger values of the snr.

The GPC shuts down the nonlinear part and performs as the linear SVM. This is the same effect that we saw for large snr in Figure 3, the training set is not long enough to ensure it can train the nonlinear part of its covariance function and it consequently sets it to zero. In Figure 4 for snr less than 10dB, although we can barely notice it, the GPC equalizer follows the nonlinear solutions, as the training sequence is long enough to train its nonlinear component in this case.

The linear SVM and GPC are 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. 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 of interests between 10 and 20dB, the GPR receivers (and nonlinear SVM) are significantly better than the linear methods and the GPC.

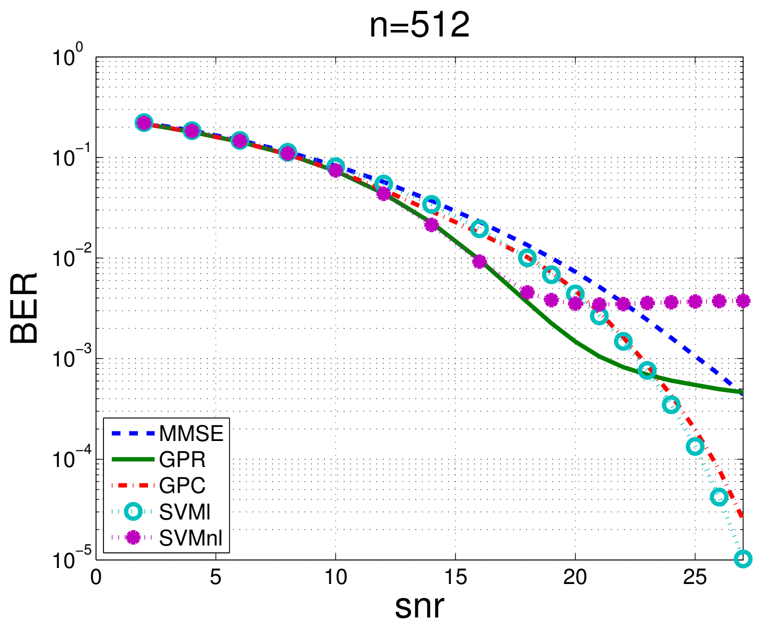


Figure 4: We report the BER versus snr for a channel equalization problem with a nonlinear channel model. The dashed line represents the linear MMSE receiver, the solid line the GPR, the dash-dotted line the GPC, the dotted line with circles the linear SVM and the dotted line with bullets the nonlinear SVM.

For this range of snr the BER is not low enough for most digital communication applications, but we can significantly reduce the BER using channel coding strategies [18] with high data rates, instead of increasing the snr.

6.4 Discussion

In the experiments we show the behavior of GPR for designing digital communication receivers and we show it has many favorable properties for solving such task when we use it with the covariance function in (28):

- If the solution is linear, the GPR receiver shuts down the nonlinear part of the covariance function and performs as the linear MMSE detector for long training sequences. It converges faster than the MMSE detector to the optimal solution. It does not degrade its performance when canceling the nonlinear part of the kernel.

- If the solution is nonlinear, the GPR receiver is able to achieve very good performances, comparable to a nonlinear SVM receiver with optimal hyperparameters, and it needs shorter training sequences to achieve such solutions. The GPR receiver performs significantly better than the linear detectors.

- The GPR receiver performs a single optimization procedure. This is a highly desirable quality as in one step we get the optimal hyperparameters without needing to try several solutions and check which one is best. The GPR decides if it needs a linear or a nonlinear solution in that single optimization without relying on a 'genie' or another procedure to check if the optimal solution is linear.

- The GPR can overfit if the training sequence is not sufficiently long, as we can see in Figure 4. But in this case the overfitting does not degrade the solution as much as it does for the nonlinear SVM. It only happens for very large snr in which we do not typically transmit.

- The GPR receiver uses a least square lost function, which is not ideal for solving classification problems when we are interested in minimizing the misclassification error. But for digital communication problems in which the noise is Gaussian, the use of this loss-function is not critical and the GPR-receiver performs as well as the receivers based on classification loss-functions (GPC and SVM).

The GPC would initially seems like a better choice for designing digital communication receivers, because it minimizes the misclassification error and it can optimize the hyperparameters, just as the GPR does. But in our experiments we show that GPC receivers usually need longer training sequences before it can tune its nonlinear part and it decides to train a linear detector in cases where a nonlinear detector clearly performs better. We believe that in order for GPC to perform better than (as well as) GPR receivers, we need far longer training sequences, which might not be available in digital communication systems. We conjecture that this limitation of GPC for training digital communication receiver is due to the posterior approximation, because its loss-function is more suitable than the ones the GPR uses and we train the GPC receiver with the same covariance function.

The SVM performs as well as GPR for the proposed problem, but it needs longer training sequence to deal with its fixed hyperparameters or longer training resources to fine-tune its hyperparameters. We do not believe there is an intrinsic advantage for GPR for this problem. Although we believe that GPR being able to tune its hyperparameters by maximum likelihood allows solving the problem easier, as we build the receiver with a single optimization procedure.

7 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. But GPR presents several properties that we believe make it a much better candidate for designing these receivers. First of all, 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. Second, its solution is analytical given the nonlinear function, while most machine learning methods need to perform an optimization problem to achieve their solution. Third, it can train its hyperparameters by maximum likelihood, while others machine learning algorithms need to cross validate their hyperparameters or structure. Forth, its computation complexity is not a limiting issue as addressed in [29].

To highlight the advantages of GPs as digital communications receivers we compare their performances to that of SVM. SVM provides solutions as good as the GPR does, but it needs more training samples. The GPR fits its covariance function by maximum likelihood, and hence it does not suffer from this problem.

The GPC could be initially thought of as a better candidate for designing digital communication receivers, since we are solving a classification problem. However, as we have shown in this paper it needs significantly longer training sequences to provide the same accuracy level as GPR receivers. One possible advantage of GPC compared to GPR for digital communication receivers is that they provide posterior probability estimates for the received bits, which could be sequentially used by a channel decoder to improve the BER. Some preliminary results of this idea can be found in [25].

References

[1] J. Arenas-Garcia, M. Martinez-Ramon, A. Navia-Vázquez, and A. R. Figueiras-Vidal. Plant identification via adaptive combination of transversal filters. *Signal Processing*, 86:2430–2438, 2006.

[2] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[3] S. Caro, F. Pérez-Cruz, and J.J. Murillo-Fuentes. Gaussian processes for regression in channel equalization. In *EUSIPCO*, Florence, Italy, September 2006.

[4] P. R. Chang and B. C. Wang. Adaptive decision feedback equalization for digital satellites channels using multilayer neural networks. *IEEE Journal on Selected areas of communication*, 13(2):316–324, February 1995.

[5] S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant. Reconstruction of binary signals using an adaptive radial-basis-function equalizer. *Signal Processing*, 22:77–83, 1991.

[6] S. Chen, A. K. Samingan, and L. Hanzo. Support vector machine multiuser receiver for DS-CDMA signals in multipath channels. *IEEE Transactions on Neural Network*, 12(3):604–611, December 2001.

[7] J. Cid-Sueiro, A. Artés-Rodríguez, and A. R. Figueiras-Vidal. Recurrent radial basis function networks for optimal symbol-by-symbol equalization. *Signal Processing*, 40:53–63, 1994.

[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, USA, 1991.

[9] D. G. M. Cruickshank. Radial basis function receivers for DS-CDMA. *IEE Electronic Letters*, 32(3):188–190, Jul 1996.

[10] G. J. Gibson, S. Siu, and C. F. N. Cowan. The application of nonlinear structures to the reconstruction of binary signals. *IEEE Transactions on Signal Processing*, 39:1877–1884, 1991.

[11] R. Gold. Optimal binary sequences for spread spectrum multiplexing. *IEEE Transactions on Information Theory*, 13(4):619–621, 1967.

[12] F. J. Gonzalez-Serrano, J. J. Murillo-Fuentes, and A. Artes-Rodriguez. GCMAC-based predistortion for digital modulations. *IEEE Transactions on Communications*, 49(9):1679–1689, September 2001.

[13] F. J. Gonzalez-Serrano, F. Perez-Cruz, and A. Artes-Rodriguez. Reduced-complexity equaliser for nonlinear channels. *Electronics Letters*, 34(9):856–858, 4 1999.

[14] Simon Haykin. *Neural Networks: A comprehensive foundation*. Prentice-Hall, 2 edition, 1999.

[15] G. S. Kimeldorf and G. Wahba. Some results in Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

[16] T. Kohonen, K. Raivio, O. Simula, O. Venta, and J. Henriksson. Combining linear equalization and self-organizing adaptation in dynamic discrete-signal detection. In *Proc. International Joint Conf. on Neural Networks*, volume 1, pages 223–228, San Diego, CA, June 1990.

[17] Malte Kuss and Carl E. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704, 10 2005.

[18] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003.

[19] M. Martinez-Ramon, J. L. Rojo-Alvarez, G. Camps-Valls, and C.G. Christodoulou. Kernel antenna array processing. *IEEE Transactions on Antennas and Propagation*, 55(3):642–650, March 2007.

[20] B. Mitchinson and R. F. Harrison. Digital communications channel equalization using the kernel adaline. *IEEE Transactions on Communications*, 50(4):571–576, 2002.

[21] Juan Jose Murillo-Fuentes, Sebastian Caro, and Fernando Pérez-Cruz. Gaussian processes for multiuser detection in CDMA receivers. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, Cambridge, MA, 2006.

[22] A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society B*, 40:1–42, 1978.

[23] Raffaele Parisi, Elio D. Di Caudio, Gianni Orlandi, and Bhaskar D. Rao. Fast adaptive digital equalization by recurent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2731–2739, November 1997.

[24] F. Pérez-Cruz and O. Bousquet. Kernel methods and their potential use in signal processing. *Signal Processing Magazine*, 21(3):57–65, 2004.

[25] F. Pérez-Cruz, P. Martínez-Olmos, and J.J. Murillo-Fuentes. Accurate posterior probability estimates for channel equalization using gaussian processes for classification. In *VIII IEEE Workshop on Signal Processing Advances in Wireless Communications*, Helsinki (Finland), June 2007.

[26] F. Pérez-Cruz and J.J. Murillo-Fuentes. Gaussian processes for digital communications. In *ICASSP*, volume V, pages 781–784, Tolousse, France, May 2006.

[27] F. Pérez-Cruz, A. Navia-Vázquez, P. L. Alarcón-Diana, and A. Artés-Rodríguez. SVC-based equalizer for burst TDMA transmissions. *Signal Processing*, 81(8):1681–1693, Aug. 2001.

[28] M. Salehi J. G. Proakis. *Communication Systems Engineering*. Prentice Hall, New York, 2 edition, 2001.

[29] Joaquin Quiñonero-Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 12 2005.

[30] G. Raleigh and J. M. Cioffi. Spatio-temporal coding for wireless communications. *IEEE Transactions on Communications*, 46(3):357–366, March 1998.

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

[32] M. P. Sanchez-Fernandez, M. de Prado-Cumplido, J. Arenas-Garcia, and F. Perez-Cruz. SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Transactions on Signal Processing*, 52(8):2298–2307, August 2004.

[33] Louis L. Scharf. *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison-Wesley, New York, 1990.

[34] B. Schölkopf and A. Smola. *Learning with kernels*. M.I.T. Press, 2001.

[35] R. Tanner and D. G. M. Cruickshank. Volterra based receivers for DS-CDMA. In *IEEE International Symposium Personal, Indoor, Mobile Radio Communications*, pages 1166–1170, September 1997.

[36] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[37] C. K. I. Williams. Prediction with Gaussian process: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 599–621. M.I.T. Press, Cambridge, (MA), 1999.

[38] C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, December 1998.

[39] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, Cambridge, MA, 1996.