Joint Nonlinear Channel Equalization and Soft LDPC Decoding with Gaussian Processes

Pablo M. Olmos, Juan José Murillo-Fuentes* *Member, IEEE* and Fernando Pérez-Cruz *Senior Member, IEEE*

***Abstract*—In this paper, we introduce a new approach for nonlinear equalization based on Gaussian processes for classification (GPC). We propose to measure the performance of this equalizer after a low-density parity-check channel decoder has detected the received sequence. Typically, most channel equalizers concentrate on reducing the bit error rate, instead of providing accurate posterior probability estimates. We show that the accuracy of these estimates is essential for optimal performance of the channel decoder and that the error rate output by the equalizer might be irrelevant to understand the performance of the overall communication receiver. In this sense, GPC is a Bayesian nonlinear classification tool that provides accurate posterior probability estimates with short training sequences. In the experimental section, we compare the proposed GPC based equalizer with state-of-the-art solutions to illustrate its improved performance.**

***Index Terms*—LDPC, SVM, Gaussian processes, equalization, machine learning, coding, nonlinear channel, soft-decoding.**

I. Introduction

In wireless communications systems, efficient use of the available spectrum is one of the most critical design issues. Thereby, modern communication systems must evolve to work as close as possible to capacity to achieve the demanded binary rates. We need to design digital communication systems that implement novel approaches for both channel equalization and coding and, moreover, we should be able to link them together to optimally detect the transmitted information.

Communication channels introduce linear and nonlinear distortions and, in most cases of interest, they cannot be considered memoryless. Inter-symbol interference (ISI), mainly a consequence of multi-path in wireless channels [1], accounts for the linear distortion. The presence of amplifiers and converters explain the nonlinear nature of communications channels [2]. Communication channels also contaminate the received sequence with random fluctuations, which are typically regarded as additive white Gaussian noise (AWGN) [3].

In the design of digital communication receivers the equalizer precedes the channel decoder. The equalizer deals with the dispersive nature of the channel and delivers a memoryless sequence to the channel decoder. The channel decoder corrects the errors at the received sequence using the controlled redundancy introduced at the transmitter. In most studies, see [4], [5], [6], [7], [2], [8], [9], [10] and the references therein, the dispersive nature of the channel and the equalizer are analyzed independently from the channel decoder. Moreover its performance gains are typically measured at very low bit error rate (BER), as if there were no channel decoder.

One of the goals of this paper is the analysis of state-of-the-art nonlinear equalizers together with the channel decoder. We make use of the fact that the equalizer performance should not be measure at low BER, but in its ability to provide accurate posterior probability estimates that can be exploited by a soft-input channel decoder to achieve capacity. Therefore measuring the performance of equalizers at low BER is meaningless, because the channel decoder can achieve those BER values at significantly lower signal power.

We employ low-density parity-check (LDPC) codes [11] to add redundancy to the transmitted binary sequence. LDPC codes have recently attracted a great research interest, because of their excellent error-correcting performance and linear complexity decoding[1]. The Digital Video Broadcasting standard uses LDPC codes for protecting the transmitted sequence and they are being considered in various applications such as 10Gb Ethernet and high-throughput wireless local area networks [13]. LDPC codes can operate with most channels of interest, such as erasure, binary symmetric and Gaussian. Irregular LDPC codes have been shown to achieve channel capacity for erasure channels [14] and close to capacity for binary symmetric and Gaussian channels [15].

For linear channels, the equalizers based on the Viterbi algorithm [16] minimize the probability of returning the incorrect sequence to the channel decoder, and they are known as maximum likelihood sequence equalizers (MLSEs). The subsequent channel decoder must treat the output of the MLSE as a binary symmetric channel, because it has no information about which bits could be in fault. Instead, we could use the BCJR algorithm [17] to design our equalizer. The BCJR algorithm returns the posterior probability (given the received sequence) for each bit, but it does not minimize the probability of returning an incorrect sequence as the Viterbi algorithm does. Nevertheless the BCJR algorithm provides a probabilistic output for each bit that can be exploited by the LPDC decoder to significantly reduce its error rate, because it has individual information about which bits might be in error. Thereby, the subsequent channel decoder substantially affects the way we measure the performance of our equalizer.

For nonlinear channels the computational complexity of the BCJR and the Viterbi algorithms grows exponentially

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).

F. Pérez-Cruz is supported by Marie Curie Fellowship 040883-AI-COM.

F. Pérez-Cruz is with the Electrical Engineering Department in Princeton University, Princeton (NJ). F. Pérez-Cruz is also an associate professor at Universidad Carlos III de Madrid (Spain). E-mail: fernando@tsc.uc3m.es

P. M. Olmos and J.J. Murillo-Fuentes are with the Dept. Teoría de la Señal y Comunicaciones, Escuela Superior de Ingenieros, Universidad de Sevilla, Paseo de los Descubrimientos s/n, 41092 Sevilla, Spain. E-mail: {olmos, murillo}@us.es

[1]The coding complexity is almost linear as proven in [12].

with the number of transmitted bits at each encoded block (frame) and they require perfect knowledge of the channel. Neural networks and, recently, machine-learning approaches have been proposed to approximate these equalizers at a lower computational complexity and they can be readily adapted for nonlinear channels. An illustrative and non-exhaustive list of examples for nonlinear equalizers are: multi-layered perceptrons [4]; radial basis functions (RBFs) [5]; recurrent RBFs [6]; wavelet neural networks [7]; kernel adaline [2]; support vector machines [8]; self-constructing recurrent fuzzy neural network [9]; and, Gaussian processes for regression [10]. But, as mentioned earlier, these approaches only compare performance at low BER without considering the channel decoder.

The aforementioned equalizers are designed to minimize their BER by undoing the effect of the channel: multi-path and nonlinearities. But their outputs cannot be directly interpreted as posterior probability estimates, which significantly limit the performance of soft-inputs channel decoders, such as LDPC codes. In this paper, we propose a channel equalizer based on Gaussian processes for classification (GPC). GPC are Bayesian machine-learning tools that assign accurate posterior probability estimates to its binary decisions, as the BCJR algorithm does for linear channels. GPC can equalize linear and nonlinear channels using a training sequence to adjust its parameters and it does not need to know a priori the channel estate information.

In a previous paper [10], we have shown that equalizers based on GPC are competitive with state-of-the-art solutions, when we compare performances at low bit error rate. In this paper, we focus on their performance after the sequence has been corrected by an LDPC code. The ability of GPC to provide accurate posterior probability predictions boosts the performance of these equalizers compared to the state-of-the-art solutions, based on support vector machines (SVMs). SVM does not provide posterior probability estimates and its output needs to be transformed, before it can be interpreted as posterior probabilities.

The transformation of SVM output into posterior probabilities has been proposed by Platt in [18] and Kwok in [19], among others. Platt's method squashes the SVM soft-output through a trained sigmoid function to predict posterior probabilities. Platt's method is not very principled, as Platt explains himself in [18], but in many cases of interest it provides competitive posterior probability predictions. In [19], the SVM output is moderated by making use of a relationship between SVM and the evidence framework for classification networks, proposed by MacKay in [20]. The moderated output can be taken as an approximation to the posterior class probability. Nevertheless, these are interpretations of the SVM output as posterior probabilities, which was not designed to provide such information [21].

The rest of the paper is organized as follows. Section II is devoted to introducing Gaussian processes. We present the receiver scheme in Section III together with the channel model and the transmitter. Also, we briefly describe the Sum Product algorithm for BCJR equalization and LDPC decoding. The application of GPC to construct an equalizer that provides probabilistic inputs to the channel decoder is developed in Section IV. In Section V, we include illustrative experiments to compare the performance of the proposed equalizers. We conclude in Section VI with some final comments.

II. Gaussian Processes for Machine Learning

Gaussian processes for machine learning are Bayesian nonlinear detection and estimation tools that provide point estimates and confidence intervals for their predictions. We specifically refer to Gaussian process for classification (GPC) for detection problems and Gaussian process for regression (GPR) for its estimation counterpart. GPR were first proposed in 1996 [22]. GPR are characterized by an analytic solution given its covariance matrix and we can estimate this covariance matrix from the data. They were subsequently extended for classification problems in [23], [24]. We have shown that GPR and GPC can be successfully applied to address the channel equalization problem [25], [10].

A. Gaussian Processes for Regression

Gaussian processes for regression is a Bayesian supervised machine learning tool for predicting the posterior probability of the output (b_*) given an input ($\mathbf{x}_*$) and a training set ($\mathcal{D} = \{\mathbf{x}_i, b_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$ $b_i \in \mathbb{R}$, i.e.

$$p(b_*|\mathbf{x}_*, \mathcal{D}). \tag{1}$$

GPR assumes that a real-valued function, known as *latent function*, underlies the regression problem and that this function follows a Gaussian process. Before the labels are revealed, we assume this latent function has been drawn from a zero-mean Gaussian process prior with its covariance function given by $k(\mathbf{x}, \mathbf{x}')$. The covariance function, also denoted as kernel, describes the relations between each pair of points in the input space and characterizes the functions that can be described by the Gaussian process. For example, $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ only yields linear latent functions and it is used to solve Bayesian linear regression problems. A detailed description of covariance functions for Gaussian processes is detailed in [26, Chap. 4].

For any finite set of input samples, a Gaussian process becomes a multidimensional Gaussian defined by its mean (zero in our case) and covariance matrix. Our Gaussian process prior becomes:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(0, \mathbf{K}), \tag{2}$$

where $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)]^\top$, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ and $(\mathbf{K})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \forall\, \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}$.

Once the labels are revealed, $\mathbf{b} = [b_1, b_2, \ldots, b_n]^\top$, together with the location of the (to-be-estimated) test point, $\mathbf{x}_*$, we can compute (1) using the standard tools of Bayesian statistics: Bayes rule, marginalization and conditioning.

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

$$p(\mathbf{f}, f(\mathbf{x}_*)|\mathcal{D}, \mathbf{x}_*) = \frac{p(\mathbf{b}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}, f(\mathbf{x}_*)|\mathbf{X}, \mathbf{x}_*)}{p(\mathbf{b}|\mathbf{X})}, \tag{3}$$

where $\mathcal{D} = \{\mathbf{b}, \mathbf{X}\}$, the probability $p(\mathbf{f}, f(\mathbf{x}_*)|\mathbf{X}, \mathbf{x}_*)$ is the Gaussian process prior in (2) extended with the test input,

$p(\mathbf{b}|\mathbf{f}, \mathbf{X})$ is the likelihood for the latent function at the training set, and $p(\mathbf{b}|\mathbf{X})$ is the evidence of the model, also known as the partition function, which guarantees that the posterior is a proper probability density function.

A factorized model is used for the likelihood function:

$$p(\mathbf{b}|\mathbf{f}, \mathbf{X}) = \prod_{i=1}^{n} p(b_i|f(\mathbf{x}_i), \mathbf{x}_i), \tag{4}$$

because the training samples have been obtained independently and identically distributed (iid). We assume that the labels are noisy observations of the latent function, $b_i = f(\mathbf{x}_i) + \nu$, and that this noise is Gaussianly distributed. The likelihood yields,

$$p(b_i|f(\mathbf{x}_i), \mathbf{x}_i) = \mathcal{N}(0, \sigma_\nu^2). \tag{5}$$

A Gaussian likelihood function is conjugate to the Gaussian prior and hence the posterior in (3) is also a multidimensional Gaussian, which simplifies the computations to obtain (1). Although other observation models for the likelihood have been proposed in the literature, as discussed in [26, Section 9.3].

We can obtain the posterior density of the output in (1) for the test point by conditioning on the training set and $\mathbf{x}_*$ and by marginalizing the latent function:

$$p(b_*|\mathbf{x}_*, \mathcal{D}) = \int p(b_*|f(\mathbf{x}_*), \mathbf{x}_*) p(f(\mathbf{x}_*)|\mathcal{D}, \mathbf{x}_*) df(\mathbf{x}_*), \tag{6}$$

where[2]

$$p(f(\mathbf{x}_*)|\mathcal{D}, \mathbf{x}_*) = \int p(f(\mathbf{x}_*), \mathbf{f}|\mathcal{D}, \mathbf{x}_*) d\mathbf{f}. \tag{7}$$

We divide the marginalization in two separate equations to show the marginalization of the latent function at the training set in (7) and the marginalization of the latent function at the test point in (6). As mentioned earlier, the likelihood and the prior are Gaussians and therefore the marginalization in (6) and (7) only involve Gaussian distributions. Thereby, we analytically compute (6) using Gaussian conditioning and marginalization properties:

$$p(b_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(\mu_{b_*}, \sigma_{b_*}^2), \tag{8}$$

where

$$\mu_{b_*} = \mathbf{k}^\top \mathbf{C}^{-1} \mathbf{b}, \tag{9}$$

$$\sigma_{b_*}^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^\top \mathbf{C}^{-1} \mathbf{k}, \tag{10}$$

and

$$\mathbf{k} = [k(\mathbf{x}_1, \mathbf{x}_*), k(\mathbf{x}_2, \mathbf{x}_*), \ldots, k(\mathbf{x}_n, \mathbf{x}_*)]^\top, \tag{11}$$

$$\mathbf{C} = \mathbf{K} + \sigma_\nu^2 \mathbf{I}. \tag{12}$$

[2]Given the training data set, $\mathbf{f}$ takes values in all the $\mathbb{R}^n$ dominium as it is a vector of n samples of a Gaussian Process.

B. Gaussian Processes for Classification

GPR can be extended to solve classification problems. In this case, the labels are drawn for a finite set and, in this section, we concentrate on binary classification, i.e. $b_i \in \{0, 1\}$. For GPC we need to change the likelihood model for the observations, because they are now either 0 or 1. The likelihood for the latent function at $\mathbf{x}_i$ is obtained using a *response function* $\Phi(\cdot)$:

$$p(b_i = 1|f(\mathbf{x}_i), \mathbf{x}_i) = \Phi(f(\mathbf{x}_i)). \tag{13}$$

The response function "squashes" the real-valued latent function to an $(0, 1)$-interval that represents the posterior probability for b_i [26]. Standard choices for the response function are $\Phi(x) = 1/(1 + exp(-x))$ and the cumulative density function of a standard normal distribution, used in logistic and probit regression respectively.

The integrals in (6) and (7) are now analytically intractable, because the likelihood and the prior are not conjugated. Therefore, we have to resort to numerical methods or approximations to solve them. The posterior distribution in (3) is typically single-mode and the standard methods approximate it with a Gaussian [26]. The two standard approximations are the Laplace method or expectation propagation (EP) [27]. In [24], EP is shown to be a more accurate approximation and we use it throughout our implementation. Using a Gaussian approximation for (3) allows exact marginalization in (7) and we can use numerical integration for solving (6), as it involves marginalizing a single real-valued quantity.

C. Covariance functions

In the previous subsection we have assumed that $k(\mathbf{x}, \mathbf{x}')$ is known, but, for most problems of interest, the best covariance function is unknown, and we need to infer it from the training samples. The covariance function describes the relation between the inputs and its form determines the possible solutions the GPC can return. Thereby, the definition of the covariance function must capture any available information about the problem at hand. It is usually defined in a parametric form as function of the so-called *hyperparameters*. The covariance function plays the same role as the kernel function in SVMs [28].

If we assume the hyperparameters, θ, to be unknown, the likelihood of the data and the prior of the latent function yield $p(\mathbf{b}|\mathbf{f}, \theta, \mathbf{X})$ and $p(\mathbf{f}|\mathbf{X}, \theta)$, respectively. From the point of view of Bayesian machine learning, we can proceed as we did for the latent function, $\mathbf{f}$. First, we compute the *marginal likelihood* of the hyperparameters of the kernel given the training dataset:

$$p(\mathbf{b}|\mathbf{X}, \theta) = \int p(\mathbf{b}|\mathbf{f}, \theta, \mathbf{X}) p(\mathbf{f}|\mathbf{X}, \theta) d\mathbf{f}. \tag{14}$$

Second, we can define a prior for the hyperparameters, $p(\theta)$, that can be used to construct its posterior density. Third, we integrate out the hyperparameters to obtain the predictions. 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 computational intensive and it is not feasible for digital communications receivers. For example, Markov-Chain Monte Carlo (MCMC) methods require several hundred to several thousand samples from the posterior of θ to integrate it out. For the interested readers, further details can be found in [26].

Alternatively, we can maximize the marginal likelihood in (14) to obtain its optimal setting [22], 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. This optimization is nonconvex [29]. 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 [26] for further details.

The covariance function must be positive semi-definite, as it represents the covariance matrix of a multidimensional Gaussian distribution. A versatile covariance function that we have previously proposed to solve channel equalization problems is described by:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha_1 \exp\left(-\sum_{\ell=1}^{d} \gamma_\ell (\mathbf{x}_{i\ell} - \mathbf{x}_{j\ell})^2\right) + \alpha_2 \mathbf{x}_i^T \mathbf{x}_j + \alpha_3 \delta_{ij}, \tag{15}$$

where $\theta = [\alpha_1, \gamma_1, \gamma_2, \ldots, \gamma_d, \alpha_2, \alpha_3]$ are the hyperparameters. The first term is a radial basis kernel, also denoted as RBF or Gaussian, with a different length-scale for each input dimension. This term is universal and allows constructing a generic nonlinear classifier. Due to the symmetry in our equalization problem and to avoid overfitting, we use the same length-scale for all dimensions: $\gamma_\ell = \gamma$ for $\ell = 1, \ldots, d$. The second term is the linear covariance function. The suitability of this covariance function for the channel equalization problem has been discussed in detail in [10].

III. Communication System

In Fig. 1 we depict a discrete-time digital-communication system with a nonlinear communication channel. We transmit independent and equiprobable binary symbols $m[j] \in \{\pm 1\}$, which are systematically encoded into a binary sequence $b[j]$ using an LDPC code. The time-invariant impulse response of the channel with length n_L is given by:

$$\mathrm{h}(z) = \sum_{\ell=0}^{n_L-1} h[\ell] z^{-\ell}, \tag{16}$$

The nonlinearities in the channel, mainly due to amplifiers and mixers, are modeled by $g(\cdot)$, as proposed in [2]. Hence, the output of the communication channel is given by:

$$x[j] = g(v[j]) + w[j] = g\left(\sum_{\ell=0}^{n_L-1} b[j-\ell] h[\ell]\right) + w[j], \tag{17}$$

where $w[j]$ represents independent samples of AWGN. The receiver equalizes the nonlinear channel and decodes the received sequence. For the channel decoder we have considered an LDPC code, because it achieves channel capacity for binary erasure channels [14] and close to capacity for Gaussian channels [15]. LDPC codes are linear block codes specified by a parity check matrix $\mathbf{H}$ with a low density number of ones, hence the name of these codes.

A. Sum-Product Algorithm

A factor graph [30] represents the probability density function of the random variable $\mathbf{y} = [y_1, \ldots, y_{n_V}]^\top$, as a product of U *potential functions*:

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{k=1}^{U} \varphi_k(\mathbf{y}_k), \tag{18}$$

where $\mathbf{y}_k$ only contains some variables from $\mathbf{y}$, $\varphi_k(\mathbf{y}_k)$ is the k^{th} potential function, and Z ensures $p(\mathbf{y})$ adds to 1. A factor graph is a bipartite graph that has a variable node for each variable y_j, a factor node for each potential function $\varphi_k(\mathbf{y}_k)$, and an edge-connecting variable node y_j to factor node $\varphi_k(\mathbf{y}_k)$ if y_j is in $\mathbf{y}_k$, i.e., if it is an argument of $\varphi_k(\cdot)$.

The sum-product (SP) algorithm [31] takes advantage of the factorization in (18) to efficiently compute any marginal distribution for any y_j:

$$p(y_j) = \sum_{\mathbf{y}/y_j} p(\mathbf{y}) = \frac{1}{Z} \sum_{\mathbf{y}/y_j} \prod_{k=1}^{U} \varphi_k(\mathbf{y}_k). \tag{19}$$

where $\mathbf{y}/y_j$ indicates that sum runs for all configurations of $\mathbf{y}$ with y_j fixed.

The SP algorithm computes the marginals for all the variables in $\mathbf{y}$ by performing local computations in each node with the information being exchanged between adjacent factor and variable nodes. The marginals are computed iteratively in a finite number of steps, if the graph is cycle-free. The complexity of the SP algorithm is linear in U and exponential in the number of variables per factor [31]. If the factor graph contains cycles, the junction tree algorithm [32] can be used to merge factors and obtain a cycle free graph, but in many cases of interest, it returns a single factor with all the variables in it. We can also ignore the cycles and run the SP algorithm as if they were none, this algorithm is known as Loopy Belief Propagation. Loopy Belief Propagation only returns an approximation to $p(y_j)$, as it ignores the cycles in the graph, and in some cases it might not converge. Nevertheless, in most cases of interest its results are accurate and it is used widely in machine learning [32], image processing [33] and channel coding [34].

B. Equalization

The LDPC decoder works with the posterior probability of each transmitted bit, given the received sequence, i.e. $p(b[j]|x[1], \ldots, x[n_C])\ \forall j = \{1, \ldots, n_C\}$. The BCJR algorithm [17] (a particularization of the SP algorithm for chains as used in digital communication community) computes this posterior probability, when the channel is linear and perfectly known at the receiver.

In Fig. 2 we show the factor graph for the channel equalizer. The variable nodes $b[j]$ represent the transmitted bits that

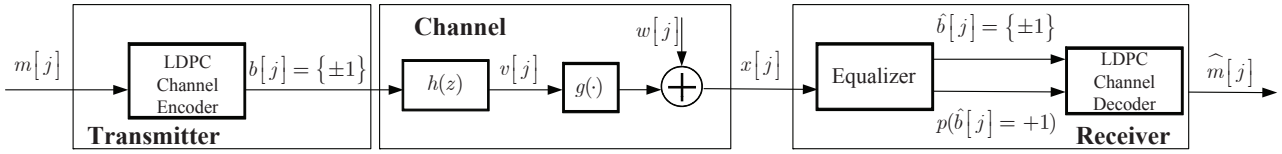


Fig. 1. Discrete-time channel model, together with the transmitter and the proposed receiver.

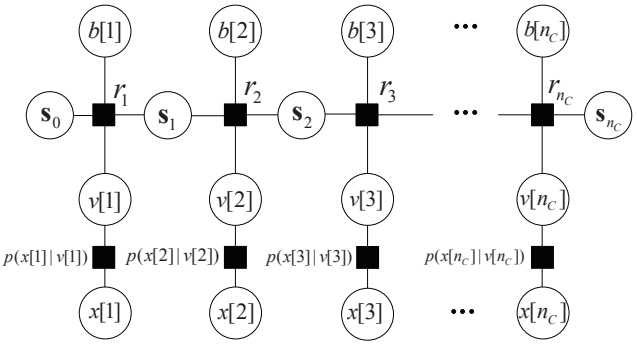


Fig. 2. Factor graph for a dispersive AWGN channel.

we want to detect, the variables nodes $x[j]$ are the observed sequence at the receiver, and the variables nodes $\mathbf{s}_j$ are the state of the channel at each time step:

$$\mathbf{s}_j = [b[j-1], \cdots, b[j-n_L+1]]^\top . \tag{20}$$

The factor nodes $p(x[j]|v[j])$ and $r_j(\cdot)$ represent, respectively, the AWGN and the dispersive nature of the channel:

$$r_j(\mathbf{s}_j, \mathbf{s}_{j-1}, b[j], v[j]) = \begin{cases} 1, & v[j] = b[j]h[0] + \mathbf{s}_{j-1}^\top \overline{\mathbf{h}} \\ 0, & \text{otherwise} \end{cases}, \tag{21}$$

where $\overline{\mathbf{h}} = [h[1], h[2], \cdots, h[n_L-1]]$. Notice that $\mathbf{s}_j$ is completely determined by $b[j]$ and $\mathbf{s}_{j-1}$.

We can run the SP algorithm, as introduced in the previous subsection, over the factor graph in Fig. 2 to obtain the desired posterior probabilities: $p(b[j]|x[1], \ldots, x[n_C])$.

C. LDPC decoding.

We have represented an example of factor graph of the dispersive channel together with the factor graph of the (n_C, k_C) LDPC code in Fig. 3. The new factors q_u, $u = 1, \ldots, n_C - k_C$, are the parity checks imposed by the LDPC code. Every parity check factor is a function of a subset of bits $i \in \mathcal{Q}_u$, the edge-connections between q_u and the bits nodes,

$$q_u = \begin{cases} 1, & \sum_{i \in \mathcal{Q}_u} b[i] \mod 2 = 0 \\ 0, & \sum_{i \in \mathcal{Q}_u} b[i] \mod 2 = 1 \end{cases}. \tag{22}$$

This factor graph contains cycles and the application of the Loopy Belief Propagation algorithm only provides posterior probability estimates for $b[j]$. For simplicity, we schedule the Loopy Belief Propagation messages in two phases. First, we run the BCJR algorithm over the dispersive noisy channel and get exact posterior probabilities. Second, we run the Loopy

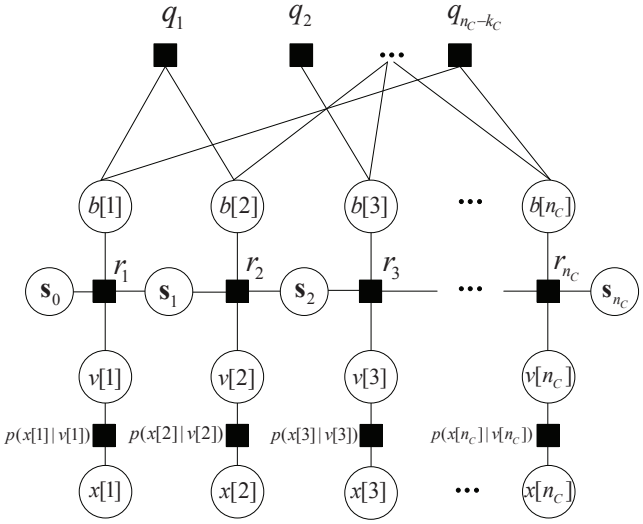


Fig. 3. Example of a joint factor graph for a dispersive AWGN channel and the LDPC decoder.

Belief Propagation over the LDPC part of the graph to correct the errors introduced by the channel and get a new estimate for the posterior probabilities of the message bits. We could then rerun the BCJR algorithm to obtain better predictions and repeat the process until convergence. But the LDPC decoding for large channel codes typically returns extreme posterior probability estimates and it is unnecessary to rerun the BCJR part of the graph, because it does not change them.

IV. Nonlinear channels: a GPC based approach

A. Probabilistic Equalization with GPC

For nonlinear channels we cannot run the BCJR algorithm to obtain the posterior probabilities $p(b[j]|x[1], \ldots, x[n_C])$; since we need an estimation of the channel, the complexity grows exponentially with the number of transmitted bits and, in most cases of interest, it cannot be computed analytically. In this paper, we propose to use GPC to accurately estimate these probabilities as follows. We first send a preamble with n bits that it is used to train the GPC as explained in Section II. Then we estimate the posterior probability for each bit

$$b_j = b[j-\tau], \tag{23}$$

using the GPC solution:

$$p(b[j-\tau]|x[1], \ldots, x[n_C]) \approx p(b_j|\mathbf{x}_j, \mathcal{D}), \quad \forall j = 1, \ldots, n_C, \tag{24}$$

with d consecutive received symbols as input vector,

$$\mathbf{x}_j = [x[j], x[j-1], \ldots, x[j-d+1]]^\top , \tag{25}$$

where d and τ represents, respectively, the order and delay of the equalizer. This is a standard solution to equalize nonlinear channels, as detailed in the Introduction. But, as far as we know, none of these proposals consider probabilistic outputs at the equalizer and its use at the decoder end.

Finally, we feed these estimates into the LDPC factor graph and iterate until all parity checks are met. The procedure is identical to the one mentioned in the previous subsection, replacing the unavailable posterior predictions by the BCJR by the GPC estimates.

B. The SVM approach

We can also follow a similar approach with other well-known nonlinear tools. For example, we can use a SVM, which is a state-of-the-art tool for nonlinear classification [35], and the methods proposed by Platt and Kwok, respectively, in [18], [19] to obtain posterior probability estimates out of the SVM output. These approaches are less principled than GPC and, as we show in the experimental section, their performances after the channel decoder are significantly worse. Furthermore, the SVM is limited in the number of hyperparameters that it can adjust for short training sequences, as discuss in [25].

C. Computational complexity

The complexity of training an SVM for binary classification is $\mathcal{O}(n^2)$, using the sequential minimal optimization [36], and Platt's and Kwok's methods add a computational complexity of $\mathcal{O}(n^2)$. The computational complexity of making predictions with SVM is $\mathcal{O}(n)$. The computational load of training the GPC grows as $\mathcal{O}(n^3)$, because we need to invert the covariance matrix. The computational complexity of making predictions with GPC is $\mathcal{O}(n2)$ [26]. In this paper, we use the full GPC version because the number of training samples was low, but there are several methods in the literature that reduced the GPC training and testing complexity to $\mathcal{O}(n)$, using a reduced set of samples [37], [38]. Therefore, the complexity of SVM and GPC are similar if we make used of these advanced techniques.

V. Experimental Results

In this section, we illustrate the performance of the proposed joint equalizer and channel decoder to show that an equalizer that provides accurate posterior probability estimates boots the performance of the channel decoder. Thereby, we should measure the ability of the equalizers to provide accurate posterior probability estimates, not only their capacity to reduce the BER.

Throughout our experiments, we use a $1/2$-rate regular LPDC code with 1000 bits per codeword and 3 ones per column and we have a single dispersive channel model:

$$\mathrm{h}(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}. \qquad (26)$$

This channel was proposed in [2] for modeling radio communication channels. In the experiments, we use 200 training samples and a four-tap equalizer ($d = 4$). The reported BER and frame error rate (FER) are computed using 10^5 test codewords and we average the results over 100 independent trials with random training and test data.

For the GPC, we use the kernel proposed in Section II-C and we have set the hyperparameter by maximum likelihood. For the SVM we use a Gaussian kernel [28] and its width and cost are trained by 10-fold cross-validation [39]. We cannot train the SVM with the kernel in (15), because it has too many hyperparameters to cross-validate. At the end of the first experiment, we also used the Gaussian kernel for the GPC to compare its performance to the SVM with the same kernel. We also use the BCJR algorithm with perfect knowledge of the channel state information, as the optimal baseline performance for the linear channel. For the nonlinear channel, the BCJR complexity makes it impractical.

In what follows, we label the performance of the joint equalizer and channel decoder by GPC-LDPC, SVM-Platt-LDPC, SVM-Kwok-LDPC and BCJR-LDPC, in which Platt and Kwok represent the method used to transform the SVM outputs into probabilities. The BER performance of the equalizers is labeled by GPC-EQ, SVM-Platt-EQ, SVM-Kwok-EQ and BCJR-EQ.

A. Experiment 1: BPSK over linear multipath channel

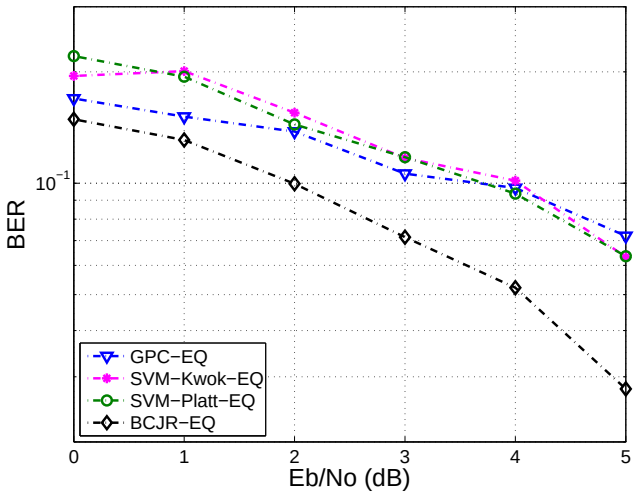
In this first experiment we deal with the linear channel in (26) and we compare our three equalizers with the BCJR algorithm with perfect channel estate information at the receiver.

In Fig. 4(a) we compare the BER of the different equalizers and in Fig. 4(b) we depict the BER measured after the channel decoder. We use throughout this section dash-dotted and solid lines to represent, respectively, the BER of the equalizers and the BER after the channel decoder.

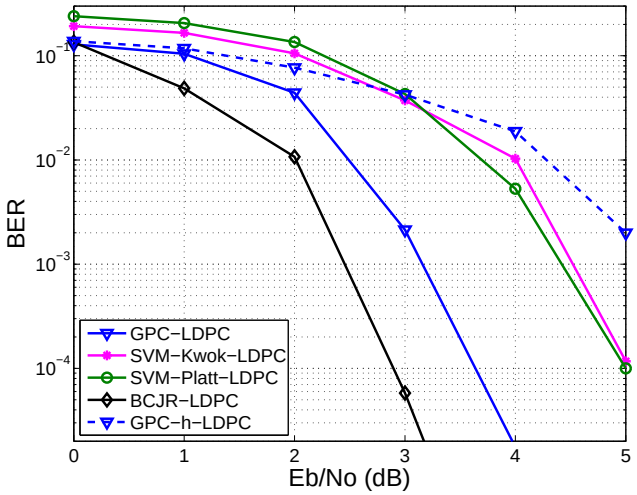
The GPC-EQ ($\triangledown$), SVM-Platt-EQ ($\circ$) and SVM-Kwok-EQ ($*$) BER plots in Fig. 4(a) are almost identical and they perform slightly worse than the BCJR-EQ ($\diamond$). These results are similar to the ones reported in [10], once the training sequence is sufficiently long. They also hold for higher normalized signal to noise ratio (E_b/N_0).

In Fig. 4(b) we can appreciate that, although the decisions provided by the GPC-EQ, SVM-Platt-EQ and SVM-Kwok-EQ are similar to each other, their estimate of the posterior probability are quite different. Therefore, the GPC-LDPC significantly reduces the BER at lower E_b/N_0, because GPC-EQ posterior probability estimates are more accurate and the LDPC decoder can rely on these trustworthy predictions. Moreover, the GPC-LDPC is less than 1dB from the optimal performance achieved by the BCJR-LDPC receiver with perfect channel information. The other receivers, SVM-Platt-LDPC and SVM-Kwok-LDPC, are over 2dB away from the optimal performance. Since both methods for extracting posterior probabilities out of the SVM output exhibit a similar performance, in what follows we only report results for the SVM using Platt's method for clarity purposes.

In Fig. 4(b) we have also included the BER performance of the GPC-h-LDPC (dashed line), whose inputs are the hard decisions given by the GPC-EQ. For this receiver, the LDPC does not have information about which bits might be in error and it has to treat each bit with equal suspicion. The BER

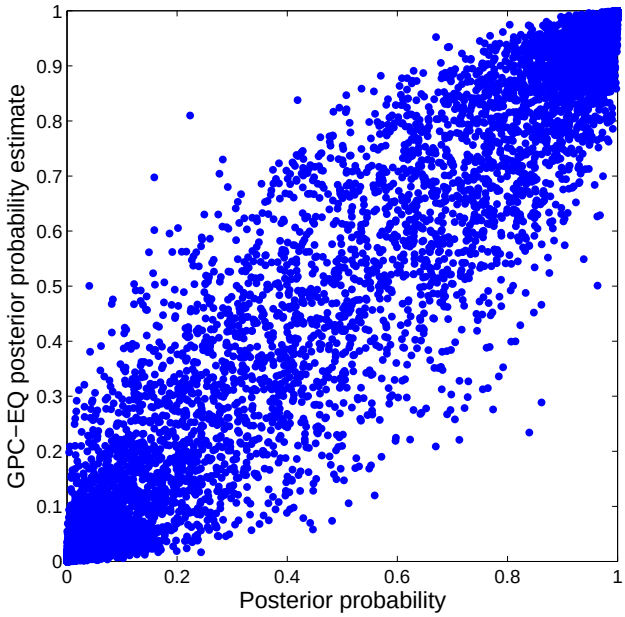


(a)

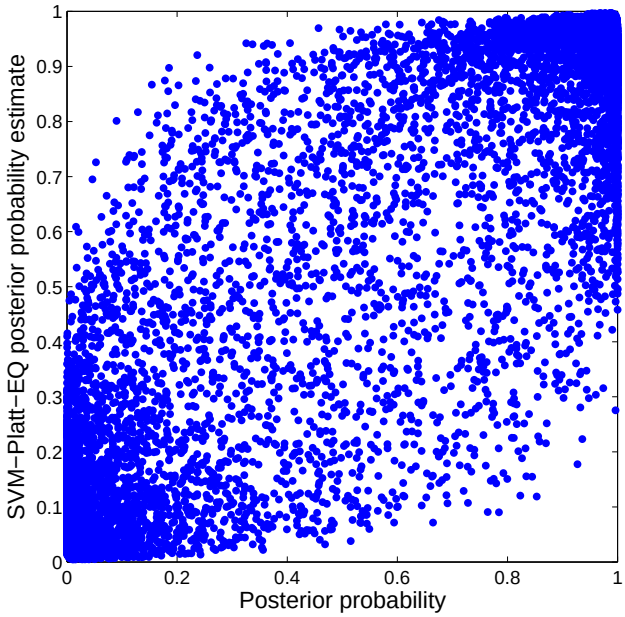


(b)

Fig. 4. We plot the BER performance for the linear channel in (26) measure at the output the equalizers in (a) and measured at the channel decoder in (b). We use dashed-dotted lines for the equalizer BER, solid lines for the LDPC BER with soft-inputs and dashed lines for the LDPC BER with hard-inputs. We represent the BCJR with $\diamond$, the GPC with $\triangledown$, the SVM with Platt's method with $\circ$ and SVM with Kwok's method with $*$.



(a)



(b)

Fig. 5. We plot the GPC-EQ and the SVM-Platt-EQ calibration curves, respectively, in (a) and (b) for $E_b/N_0 = 2$dB.

performance of SVM-Kwok-h-LDPC and SVM-Platt-h-LDPC are similar to GPC-h-LDPC and they are not shown to avoid cluttering the figure. It can be seen that even though the posterior probability estimates of Platt's and Kwok's methods are not as accurate as GPC, they are better than not using them at all.

To understand the difference in posterior probability estimates, we have plotted calibration curves for the GPC-EQ and SVM-Platt-EQ, respectively, in Fig. 5(a) and Fig. 5(b) for $E_b/N_0 = 2$ dB. We depict the estimated probabilities versus the true ones. We can appreciate that the GPC-EQ posterior probability estimates are closer to the main diagonal and they are less spread. Thereby GPC-EQ estimates are closer to the true posterior probability, which explains its improved performance with respect to the SVM-Platt-EQ, when we measure the BER after the LDPC decoder.

To complete this first experiment we have also trained a GPC equalizer with the Gaussian kernel used for the SVM. We have plot its BER (GPC-LDPC-Gauss, $\square$) after the channel decoder in Fig. 6 alongside with the GPC-LDPC and SVM-Platt-LDPC solutions. From this plot we can understand that the improved performance of the GPC with respect to the SVM is based on both its ability to provide accurate posterior probability estimates and its ability to train a more versatile kernel. In the remaining experiments, we use the versatile kernel for GPC in (15), as it is an extra feature of GPs compared to SVMs.

We have shown that GPC-LDPC is far superior to the

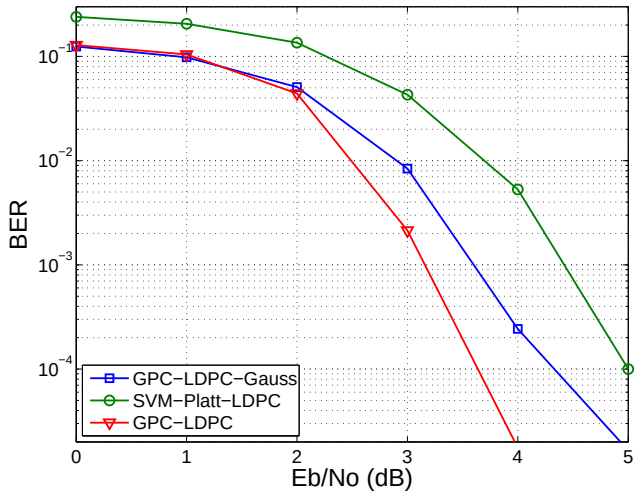


Fig. 6. We plot the BER performance at the output of the LDPC decoder with soft-inputs using a GPC equalizer with Gaussian kernel (□), a SVM equalizer with Platt's method and Gaussian kernel (○), and a GPC equalizer with the kernel in (15) (▽).

other schemes and its performance is close to optimal. This result shows that using a method that can predict accurately the posterior probability estimates allows the LDPC decoding algorithm to perform to its fullest. From this first experiment, it is clear that we need to compare the equalizers performance after the channel decoder, otherwise the BER measured after the equalizers do not tell the whole story. Also, we do not need to compare the equalizers at low BER, because the channel decoder reduces the BER significantly.

B. Experiment 2: Nonlinear multipath channel 1

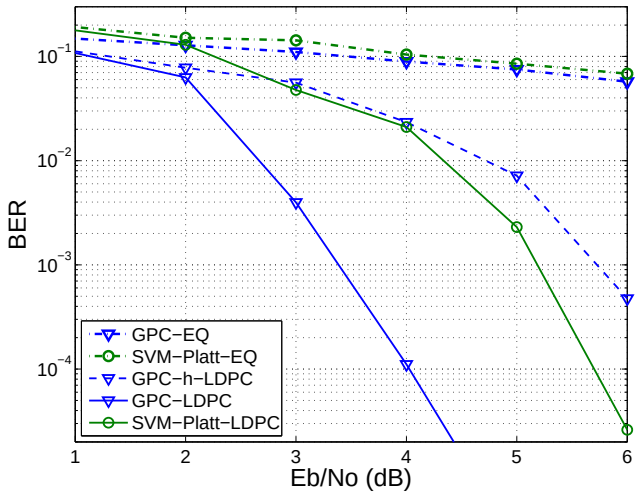
In the next two experiments we face nonlinear multipath channels. We assume the nonlinearities in the channel are unknown at the receiver and transmitter, and we need a nonlinear equalizer, which is able to compensate the nonlinearities in the channel. For this experiment we use the channel model proposed in [2], [9]:

$$|g(v)| = |v| + 0.2|v|^2 - 0.1|v|^3. \tag{27}$$

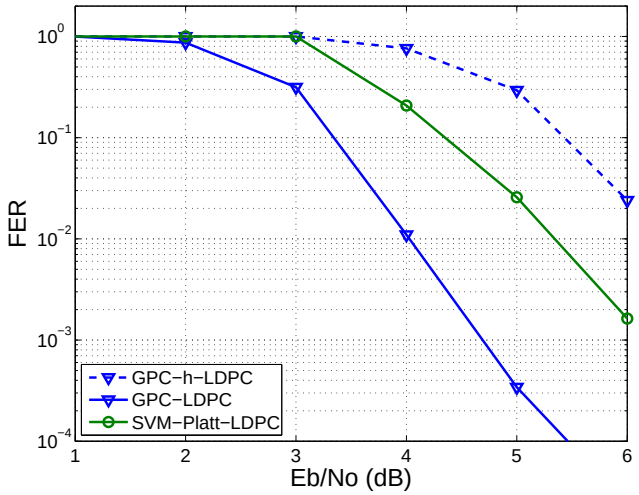
This model represents an amplifier working in saturation with 0 dB of back off, which distorts the amplitude of our modulated BPSK signal.

In Fig. 7(a) we compare the BER performance of the GPC-LDPC (▽) with the SVM-Platt-LDPC (○). We also plot for completeness the BER after the equalizer with dash-dotted lines for the two compared receivers. For this channel model the BCJR is unavailable, since its complexity is exponential in the length of the encoded sequence.

The two equalizers perform equally well, while the BER by the GPC-LDPC is significantly lower than that of SVM-Platt-LDPC. Again, the ability of the GPC to return accurate posterior probability estimates notably improves the performance of the channel decoder. In this example, the coding gain is about 2 dB between the GPC-LDPC and the other equalizers. Also the BER of the LDPC with hard outputs (GPC-h-LPDC) is higher than the soft-outputs receivers. This result is relevant



(a)



(b)

Fig. 7. Performance at the output of the equalizers (dash-dotted lines) and the channel decoder with soft-inputs (solid lines) and with hard-inputs (dashed line) for the GPC (▽) and the SVM-Platt (○). The BER is included in (a) while the FER is depicted in (b), for the channel in (26) with the nonlinearities in (27).

because, even though our posterior probability estimates are not quite accurate, we are better off with them than without.

For completeness, we have also depicted the FER in Fig. 7(b). The FER performance is typically used when we are only interested in error-free frames, which is a more relevant measure in data-package networks. The results in FER are similar to the BER. The coding gain between the GPC-LDPC receiver and the SVM-Platt-LDPC is around 1 dB, instead of 2. This difference in performance can be explained, as well, by the posterior probability estimates given by the GPC. For the frames that cannot be decoded by the LPDC, the GPC posterior probabilities are more accurate and allow the LPDC to return a frame with fewer bits in error.

C. Experiment 3: Nonlinear multipath channel 2

To conclude our experiments, we consider a second model for a nonlinear amplifier working in saturation. This model

was proposed in [9], [40], [41]:

$$|g(v)| = |v| + 0.2|v|^2 - 0.1|v|^3 + 0.5\cos(\pi|v|), \qquad (28)$$

and considers an additional term that further complicates the performance of the equalizer.

In Fig. 8 we have depicted the BER and FER for the GPC and SVM equalizers. The performance of the two receivers follows the same lines that we have seen in the previous cases: the equalizers perform equally well, while GPC-LPDC outperforms the SVM receiver. The main difference in this experiment is the threshold behavior that the SVM-Platt-LDPC exhibit. For E_b/N_0 lower than 5 dB their BERs are worse than the GPC-h-LDPC BER, which means that their posterior probability estimate are way off and we would be better off without them. But once the E_b/N_0 is large enough their posterior probability estimates are sufficiently accurate and they outperform the LPDC decoder with hard outputs. From the FER point of view, the soft-output equalizers outperform the hard equalizer for all E_b/N_0 ranges. Thereby, the use of soft-output equalizers is always favorable, even in strong nonlinear channels, in which the posterior probability estimates might not be accurate.

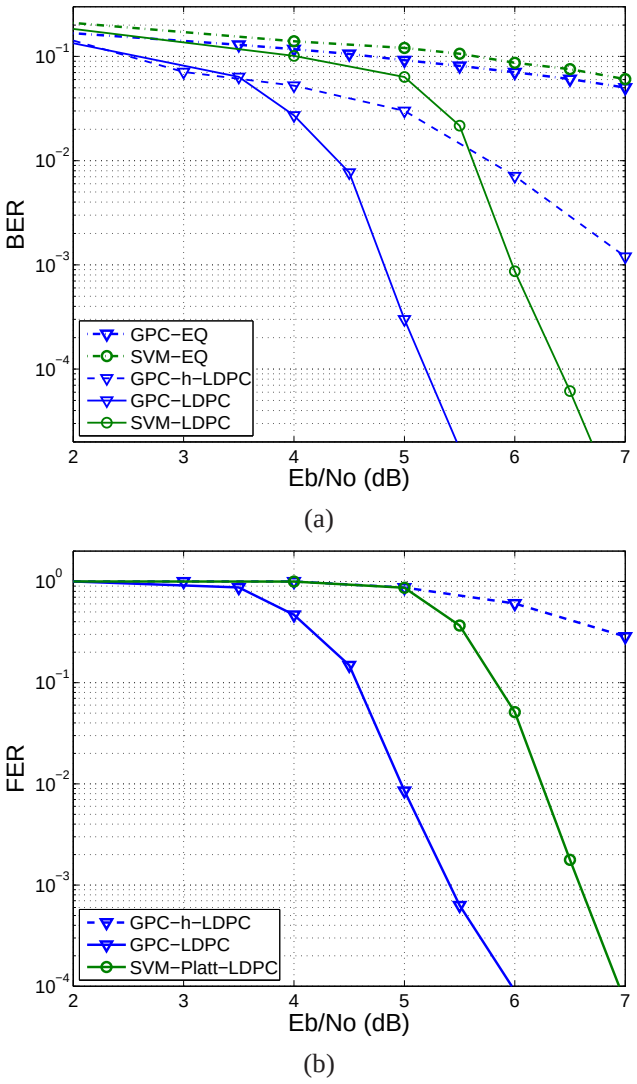


Fig. 8. Performance at the output of the equalizers (dash-dotted lines) and the channel decoder with soft-inputs (solid lines) and with hard-inputs (dashed line) for the GPC ($\triangledown$) and the SVM-Platt ($\circ$). The BER is included in (a) while the FER is depicted in (b), for the channel in (26) with the nonlinearities in (28).

VI. Conclusions

The probabilistic nonlinear channel equalization is an open problem, since the standard solutions such as the nonlinear BCJR exhibit exponential complexity with the length of the encoded sequence. Moreover, they need an estimation of the nonlinear channel and they only approximate the optimal solution [42]. In this paper, we propose GPC to solve this long-standing problem. GPC is a Bayesian nonlinear probabilistic classifier that produces accurate posterior probability estimates. We compare the performance of the different probabilistic equalizers at the output of an LDPC channel decoder. We have shown the GPC outperforms the SVM with probabilistic output, which is a state-of-the-art nonlinear classifier.

Finally, as a by-product, we have shown that we need to measure the performance of the equalizers after the channel decoder. The equalizers' performance is typically measured at low BER without considering the channel decoder. But if we do not incorporate the channel decoder the BER output by the equalizer might not give an accurate picture of its performance. Furthermore, the equalizer performance at low BER might not be illustrative, as the channel decoder significantly reduces the BER for lower signal to noise ratio values.

References

[1] J. G. Proakis and D. G. Manolakis, *Digital Communications.* Prentice Hall, 2007.

[2] B. Mitchinson and R. F. Harrison, "Digital communications channel equalization using the kernel adaline," *IEEE Transactions on Communications*, vol. 50, no. 4, pp. 571–576, 2002.

[3] J. G. Proakis and M. Salehi, *Communication Systems Engineering*, 2nd ed. New York: Prentice Hall, 2002.

[4] S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant, "Adaptative equalization of finite non-linear channels using multilayer perceptrons," *Signal Processing*, vol. 10, pp. 107–119, 1990.

[5] ——, "Reconstruction of binary signals using an adaptiveradial-basis-function equalizer," *Signal Processing*, vol. 22, pp. 77–83, 1991.

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

[7] 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*, vol. 13, no. 2, pp. 316–324, Feb. 1995.

[8] F. Pérez-Cruz, A. Navia-Vázquez, P. L. Alarcón-Diana, and A. Artés-Rodríguez, "SVC-based equalizer for burst TDMA transmissions," *Signal Processing*, vol. 81, no. 8, pp. 1681–1693, Aug. 2001.

[9] T. Kohonen, K. Raivio, O. Simula, O. Venta, and J. Henriksson, "Design of an SCRFNN-based nonlinear channel equaliser," in *IEEE Proceedings on Communications*, vol. 1552, Banff, Canada, Dec. 2005, pp. 771–779.

[10] F. Pérez-Cruz, J. Murillo-Fuentes, and S. Caro, "Nonlinear channel equalization with Gaussian Processes for Regression," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 5283–5286, Oct. 2008.

[11] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[12] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 72, no. 2, pp. 638–656, 2001.

[13] H. Zhong and T. Zhang, "Block-LDPC: A practical LDPC coding system

design approach," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 4, 2005.
[14] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *29th Annual ACM Symposium on Theory of Computing*, 1997, pp. 150 – 159.
[15] S. Chung, D. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, 2001.
[16] D. Forney, "The Viterbi algorithm," *IEEE Proceedings*, vol. 61, no. 2, pp. 268–278, Mar. 1973.
[17] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
[18] J. C. Platt, "Probabilities for SV machines," in *Advances in Large Margin Classifiers*, A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. Cambridge, (MA): M.I.T. Press, 2000, pp. 61–73.
[19] J. Kwok, "Moderating the outputs of support vector machine classifier," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1018–1031, Sept. 1999.
[20] D. MacKay, "The evidence framework applied to classification networks," *Neural Computation*, vol. 4, no. 5, pp. 720–736, 1992.
[21] V. N. Vapnik, *Statistical Learning Theory*. New York: John Wiley & Sons, 1998.
[22] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996, pp. 598–604.
[23] C. K. I. Williams and D. Barber, "Bayesian classification with Gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1342–1351, Dec. 1998.
[24] M. Kuss and C. Rasmussen, "Assessing approximate inference for binary Gaussian Process classification," *Machine learning research*, vol. 6, pp. 1679–1704, Oct. 2005.
[25] F. Pérez-Cruz and J. J. Murillo-Fuentes, "Digital communication receivers using Gaussian processes for machine learning," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, 2008.
[26] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
[27] T. Minka, "Expectation propagation for approximate bayesian inference," in *UAI*, 2001, pp. 362–369.
[28] F. Pérez-Cruz and O. Bousquet, "Kernel methods and their potential use in signal processing," *Signal Processing Magazine*, vol. 21, no. 3, pp. 57–65, 2004.
[29] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge, UK: Cambridge University Press, 2003.
[30] H. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, pp. 28–41, January 2004.
[31] F. R. Kschischang, B. I. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Inform Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[32] M. I. Jordan, Ed., *Learning in Graphical Models*. Cambridge, MA: MIT Press, 1999.
[33] E. Sudderth and W. Freeman, "Signal and image processing with belief propagation [DSP applications]," *IEEE Signal Processing Magazine*, vol. 25, pp. 114–141, March 2008.
[34] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.
[35] B. Schölkopf and A. Smola, *Learning with kernels*. M.I.T. Press, 2001.
[36] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training suppor vector machines," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, (MA): M.I.T. Press, 1999, pp. 185–208.
[37] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*. MIT press, 2006, pp. 1257–1264.
[38] L. Csató and M. Opper, "Sparse online Gaussian processes," *Neural Computation*, vol. 14, pp. 641–668, 2002.
[39] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.
[40] B. Majhi and G. Panda, "Recovery of digital information using bacterial foraging optimization based nonlinear channel equalizers," in *IEEE 1st International Conference on Digital Information Management*, Dec. 2006, pp. 367–372.
[41] J. Patra, R. Pal, R. Baliarsingh, and G. Panda, "Nonlinear channel equalization for QAM signal constellation using artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 29, pp. 262–271, April 1999.
[42] M. Mesiya, P. McLane, and L. Campbell, "Maximum likelihood sequence estimation of binary sequences transmitted over bandlimited nonlinear channels," *IEEE Transactions on Communications*, vol. 25, pp. 633–643, April 1977.