EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Support vector machines for optimal channel decoding

Gastón De Boni Rovella[1,2*], Meryem Benammar[3*], Tarik Benaddi[4] and Hugo Méric[5]

*Correspondence: gdeboni@fing.edu.uy; meryem.benammar@isae-supaero.fr

[1] TéSA Research Lab, 7 Boulevard de la Gare, 31400 Toulouse, France
[2] Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, 11300 Montevideo, Uruguay
[3] ISAE-SUPAERO, 10 Av. Marc Pélegrin, 31055 Toulouse, France
[4] Thales Alenia Space, 26 Av. Jean Francois Champollion, 31100 Toulouse, France
[5] Centre National d'Études Spatiales, 18 Av. Edouard Belin, 31400 Toulouse, France

Abstract

In this work, we investigate channel decoding techniques based on machine learning, and more specifically, on support vector machines (SVMs). Existing SVM-based decoders suffer from a scalability problem, characterized by the exponential growth of both the number of classifiers required to decode and the size of the training dataset as a function of the code dimension. This phenomenon, often referred to as the *curse of dimensionality*, renders existing SVM approaches impractical for larger codes. To partly address this limitation, we introduce a novel bit-wise SVM decoding strategy coupled with a noiseless optimization framework, which significantly reduces the computational burden. Specifically, for a code of length n and dimension k, our approach decreases the number of SVM classifiers from an exponential dependence to a linear in k, while the required training dataset is minimized to a single noiseless codeword per class, which remains exponential in k. We formulate the resulting optimization problem and derive its analytical solution. Moreover, we demonstrate that, under specific conditions, the solution of the optimization process produced by the proposed framework is equivalent to the optimal Maximum A Posteriori (MAP) decoding rule when applied to transmissions over additive white Gaussian noise (AWGN) channels.

Keywords: Channel decoding, Support vector machine, Maximum likelihood decoding, Machine learning

1 Introduction

Since the deployment of the 5 G standard, there has been an increasing interest in the beyond-5 G and 6 G communication technologies, with a particular emphasis on the usage of machine learning both at the physical layer and at the above layers of the communication protocol stack [1–3]. The introduction of machine learning aims at handling complex and time-consuming communication problems in a data-based approach, as opposed to the traditional model-based approach [4]. This alleviates the limitations of sub-optimal mathematical modeling and relegates the traditional online complexity to an offline, possibly complex, training procedure.

Of particular interest in this work is the physical layer forward error correction (FEC), and more specifically, the decoding operation at the receiver which can be described as a high-dimensional classification problem. The design of low complexity and low latency decoding solutions for a given error correction code based on machine learning dates

back to the 80 s [5–7]. However, with recent advances in computer science and computing power, their interest has increased dramatically, even though is often directed toward deep learning solutions (neural networks [8–10], attention-based networks [11], etc.). In this work, we investigate an alternative to neural networks for high-dimensional classification, namely support vector machines (SVMs).

SVMs were introduced over thirty years ago by Vapnik, Boser, Cortes, and Guyon [12–14] and present three main advantages with respect to deep neural network solutions: (i) They are maximum margin classifiers and are thus more robust to mismatches between training and application channel conditions [15]; (ii) the optimization algorithm to be solved is convex and therefore converges to a global minimum; and (iii) due to the nature of the optimization problem, and there is very little risk of overfitting. This motivated the application of SVMs to channel decoding as was undertaken in [16–18]. However, the therein *one vs. one* and *one vs. rest* suggested approaches produce a number of SVMs exponential in the block length and, hence, become quickly intractable when the latter increases to more than a few bits. This is because these methods need to produce at least one decision function per valid codeword, and the number of possible codewords increases exponentially with the size of the code.

In this work, we revisit SVM-based channel decoding by means of (i) suggesting a bit-wise approach that reduces the number of SVMs necessary for decoding from exponential to linear in the number of information bits; (ii) reducing the complexity of the optimization process by reducing the size of the training dataset to a single codeword per class; and (iii) proving that the suggested SVM bit-wise classifier is equivalent to the bit-wise Maximum A Posteriori (MAP) decoding over an additive white Gaussian noise (AWGN) channel.

The remainder of this work is organized as follows. Section 2 describes the system model and the channel decoding problem. Section 3 gives an overview of SVMs, and their application to both linearly separable and nonlinearly separable data. Section 4 describes previous efforts in the SVM-based channel decoding problem, and the two main approaches hitherto implemented. In Section 5, we introduce our solution and its advantages compared to previous works and investigate the optimality of SVM for channel decoding. In Section 6, we illustrate our results with numerical simulations and analyze the complexity and robustness of the system, and we provide a short discussion on kernel functions for SVM-based decoding. Conclusions and future directions are drawn in section 7.

1.1 Notations

Uppercase Roman and bold letters (e.g., X and $\boldsymbol{X}$) will denote random variables and vectors, and lowercase (e.g., x and $\boldsymbol{x}$) their realizations. The notations $\mathbb{P}_X(x)$, $\mathbb{P}_{X,Y}(x,y)$, and $\mathbb{P}_{X|Y}(x|y)$ represent, respectively, single, joint, and conditional pmfs (or pdfs) resp. $\mathbb{P}(\cdot)$ represents the event probability, $\mathbb{E}_X(\cdot)$ the expectation over the distribution of X, $\mathbb{1}(\cdot)$ the Boolean indicator operator, and i.i.d. means independent and identically distributed. $\mathcal{R}(y)$ and $\mathcal{I}(y)$ denote, respectively, the real and imaginary parts of $y \in \mathbb{C}$; $[\boldsymbol{a}, \boldsymbol{u}]$ denotes the concatenation of the vectors $\boldsymbol{a}$ and $\boldsymbol{u}$, while I_n is the identity matrix of size n. $|\mathcal{S}|$ denotes the cardinality of the set $\mathcal{S}$, while $[1:k]$ denotes the integers from 1 to k.

2 Problem statement

2.1 The channel decoding problem

Let us consider a coded and modulated transmission over an AWGN channel as depicted in Figure 1. In such a setting, a random binary input message $\boldsymbol{u} \in \{0,1\}^k$ consisting of k independent bits uniformly distributed is mapped through an (n, k) linear block error correction code into a codeword $\boldsymbol{c} \in \{0,1\}^n$ of n bits. The obtained codeword is mapped through a linear modulation scheme, such as phase-shift keying (PSK) or quadrature amplitude modulation (QAM) of order m, thus yielding a complex vector $\boldsymbol{x} \in \mathbb{C}^{n'}$, where $n' = n/m$. The channel output $\boldsymbol{y}$ results from the transmission of the input $\boldsymbol{x}$ through a memoryless AWGN channel, i.e., $\boldsymbol{y} \triangleq \boldsymbol{x} + \boldsymbol{w}$ where $\boldsymbol{w} \overset{\text{i.i.d}}{\sim} \mathcal{CN}(0, \sigma^2 I_{n'})$.

In order to feed real-valued vectors to the decoder, we first perform a preprocessing stage of the complex-valued channel output $\boldsymbol{y}$,

$$\tilde{\boldsymbol{y}} \triangleq [\, \mathcal{R}(\boldsymbol{y}),\ \mathcal{I}(\boldsymbol{y}) \,], \tag{1}$$

which yields a real-valued signal $\tilde{\boldsymbol{y}} \in \mathbb{R}^{2n'}$. This signal is then fed to a decision rule (decoder) $\boldsymbol{g}(\cdot)$ which produces an estimate of the original message of k bits $\hat{\boldsymbol{u}} \triangleq \boldsymbol{g}(\tilde{\boldsymbol{y}}) \in \{0,1\}^k$.

For a fixed (n, k) linear block code, *the channel decoding problem* consists in developing a decoding rule $\boldsymbol{g}(\cdot)$ that minimizes the average bit error probability (BEP) defined by

$$P_e^b \triangleq \frac{1}{k} \sum_{j=1}^{k} \mathbb{P}(\hat{U}_j \neq U_j) = \frac{1}{k} \sum_{j=1}^{k} \mathbb{P}(g_j(\boldsymbol{Y}) \neq U_j), \tag{2}$$

where U_j and $\hat{U}_j$ are binary random variables corresponding to the jth element of the random message $\mathbf{U}$ and estimated message $\hat{\mathbf{U}}$, respectively.

2.2 The bit-MAP decoding rule

The theoretically optimal decoding rule from a BEP point of view is the so-called bit-MAP decoding rule which we state in the following lemma.

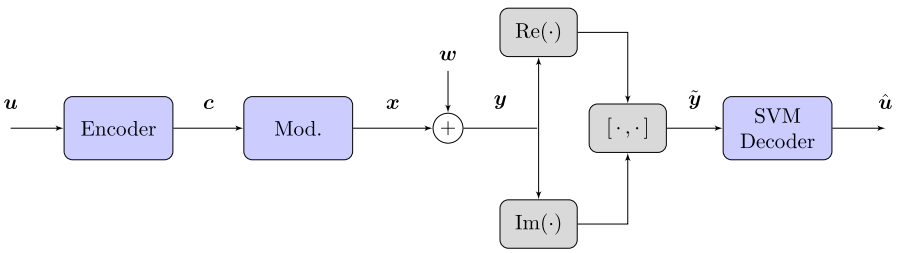

Fig. 1 System model of a simplified communication chain, including the concatenation of the real and imaginary parts for the SVM processing

Lemma 1 (Bit-MAP decoding rule) *The optimal decoding rule for the BEP defined in* (2) *is given by the concatenation of k bit-MAP decoding rules* $\{g^{(j)}(\cdot)\}_{1\leq j\leq k}$ *where, for all* $j \in [1:k]$,

$$g^{(j)}(\boldsymbol{y}) \triangleq \underset{u\in\{0,1\}}{\operatorname{argmax}}\, \mathbb{P}(U_j = u|\boldsymbol{y}). \tag{3}$$

1 ***Proof***

The proof follows from classical tools from detection theory and can be found in [19].

□

The bit-MAP decoding rule can be further simplified under the assumption of i.i.d. and uniformly distributed input bits.

Lemma 2 (Bit-MAP simplification) *Under the assumption of i.i.d. and uniformly distributed bits* $\{u_j\}_{1\leq j\leq k}$, *the bit-MAP decoding rule is equivalent to the bit maximum likelihood* (*bit-ML*) *decoder, expressed as*

$$g^{(j)}(\boldsymbol{y}) = \mathbb{1}\left(\sum_{\substack{\boldsymbol{u}\in\{0,1\}^k,\\ u_j=1}} P_{Y|X}(\boldsymbol{y}|\boldsymbol{x}(\boldsymbol{u})) > \sum_{\substack{\boldsymbol{u}\in\{0,1\}^k,\\ u_j=0}} P_{Y|X}(\boldsymbol{y}|\boldsymbol{x}(\boldsymbol{u})) \right), \tag{4}$$

where $P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})$ *denotes the channel conditional probability density function* (*pdf*) *and* $\boldsymbol{x}(\boldsymbol{u})$ *denotes the modulated codeword associated with the message* $\boldsymbol{u}$.

1 ***Proof***

Although a classical result, the proof is given for clarity of later results in the paper. Let $\boldsymbol{y}$ *be a received sequence. Let us assume* $\boldsymbol{y}$ *has a nonzero marginal pdf, i.e.,* $P_Y(\boldsymbol{y}) \neq 0$. *Then, we have by definition for all* $j \in [1:k]$,

$$g^{(j)}(\boldsymbol{y}) = \begin{cases} 1 & \text{if} \quad \dfrac{\mathbb{P}(U_j=0|\boldsymbol{y})}{\mathbb{P}(U_j=1|\boldsymbol{y})} < 1 \\ 0 & \text{else.} \end{cases} \tag{5}$$

Next, by observing that for $u \in \{0,1\}$,

$$\mathbb{P}(U_j = u|\boldsymbol{y}) = \sum_{\substack{\boldsymbol{u}\in\{0,1\}^k,\\ u_j=u}} \mathbb{P}(\boldsymbol{U}=\boldsymbol{u}|\boldsymbol{y}) \tag{6}$$

$$= \frac{2^{-k}}{P_{\boldsymbol{Y}}(\boldsymbol{y})} \sum_{\substack{\boldsymbol{u}\in\{0,1\}^k,\\ u_j=u}} P_{Y|X}(\boldsymbol{y}|\boldsymbol{x}(\boldsymbol{u})), \tag{7}$$

where we used Bayes' identity with the assumption of i.i.d. uniformly distributed input bits $\mathbb{P}(\boldsymbol{U} = \boldsymbol{u}) = 2^{-k}$, and the fact that the encoding and modulations are one-to-one mappings, i.e., $P_{\boldsymbol{Y}|\boldsymbol{U}}(\boldsymbol{y}|\boldsymbol{u}) = P_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x}(\boldsymbol{u}))$. The proof of the lemma follows from simplifying the likelihood ratio in (5) and rewriting it using an indicator operator. □

Under the bit-ML formulation in (4), the summations imply a marginalization over binary sequences of length $k-1$, rendering the complexity of the bit-ML decoder exponential in k. Hence, for generic block codes for which the marginalization cannot be done efficiently on sparse graphs or trellises, the bit-ML remains an impractical algorithm.

In this work, we will investigate the construction of SVM-based decoders and study their connection to the bit-ML decoding rule in the case of AWGN channels and uniformly distributed bits.

2.3 Methods/Experimental

Experimental results are given in Section 6, and perspectives on further applications are discussed in Section 7.

3 Preliminaries on SVM

The main idea behind SVMs consists in using a set of labeled data—each element belonging to one of two possible classes—to produce a separating hyperplane that divides the data into the two corresponding classes while maximizing the margin between the hyperplane and the two classes. For this reason, it is known as a *maximum margin classifier*. In the following, a brief introduction to SVMs is given, along with the necessary elements to support our proposed SVM-based receiver—i.e., the kernel method and the multiclass SVM.

3.1 SVM for linearly separable data

Let us consider a labeled dataset $\{\tilde{\boldsymbol{y}}_i, l_i\}_{1\leq i\leq N}$ which consists of N vectors $\tilde{\boldsymbol{y}}_i \in \mathbb{R}^{2n'}$ with their respective labels $l_i \in \{-1, +1\}$. Let the class $\mathcal{C}_0$ be the set of vectors $\tilde{\boldsymbol{y}}_i$ for which $l_i = -1$, and $\mathcal{C}_1$ the vectors for which $l_i = +1$. The dataset is said to be linearly separable if there exists $\boldsymbol{\xi} \in \mathbb{R}^{2n'}$ and $\nu \in \mathbb{R}$ that define a hyperplane $\mathcal{P} \in \mathbb{R}^{2n'}$ that satisfies the following

$$\mathcal{P} : \{\tilde{\boldsymbol{y}} \in \mathbb{R}^{2n'} \mid f(\tilde{\boldsymbol{y}}) = \boldsymbol{\xi}^T \tilde{\boldsymbol{y}} + \nu = 0\} \tag{8}$$

$$\begin{aligned} \text{such that: } & f(\tilde{\boldsymbol{y}}) < 0 \ \text{ for all } \tilde{\boldsymbol{y}} \in \mathcal{C}_0 \\ & f(\tilde{\boldsymbol{y}}) \geq 0 \ \text{ for all } \tilde{\boldsymbol{y}} \in \mathcal{C}_1. \end{aligned} \tag{9}$$

The SVM principle consists in producing a hyperplane $\mathcal{P}^*$ that satisfies the *maximum margin property*, i.e., being at an equal and maximum distance from the nearest points of each class (the so-called *support vectors*). Let $\tilde{\boldsymbol{y}}^*$ denote the closest point to the hyperplane $\mathcal{P}^*$ and belonging to any of the two classes. Without loss of generalization, we can normalize $\boldsymbol{\xi}$ and ν so that

$$|f(\tilde{\boldsymbol{y}}^*)| = |\boldsymbol{\xi}^T \tilde{\boldsymbol{y}}^* + \nu| = 1. \tag{10}$$

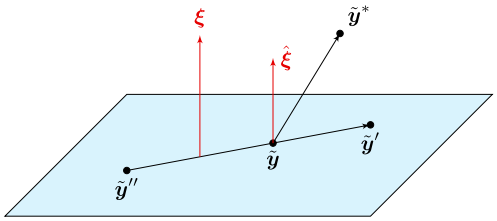

Fig. 2 Visual aid for the SVM optimization problem deduction

Let us compute the distance between $\tilde{\boldsymbol{y}}^*$ and the hyperplane (see Figure 2). Consider two points on the plane, $\tilde{\boldsymbol{y}}'$ and $\tilde{\boldsymbol{y}}''$. We have that

$$\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}' + \nu = \boldsymbol{\xi}^T\tilde{\boldsymbol{y}}'' + \nu = 0 \;\Rightarrow\; \boldsymbol{\xi}^T(\tilde{\boldsymbol{y}}' - \tilde{\boldsymbol{y}}'') = 0, \tag{11}$$

and, thus, $\boldsymbol{\xi}$ is orthogonal to the plane $\mathcal{P}$. Consider a point $\tilde{\boldsymbol{y}}$ on the plane, and the normalized vector $\hat{\boldsymbol{\xi}} = \frac{\boldsymbol{\xi}}{||\boldsymbol{\xi}||}$. In this case, the distance between $\tilde{\boldsymbol{y}}^*$ and the plane can be computed as the projection of $\tilde{\boldsymbol{y}}^* - \tilde{\boldsymbol{y}}$ on $\hat{\boldsymbol{\xi}}$:

$$d(\tilde{\boldsymbol{y}}^*, \mathcal{P}) = |\hat{\boldsymbol{\xi}}^T(\tilde{\boldsymbol{y}}^* - \tilde{\boldsymbol{y}})| \tag{12}$$

$$= \frac{1}{||\boldsymbol{\xi}||}|\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}^* - \boldsymbol{\xi}^T\tilde{\boldsymbol{y}}| \tag{13}$$

$$= \frac{1}{||\boldsymbol{\xi}||}|\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}^* + \nu - (\underbrace{\boldsymbol{\xi}^T\tilde{\boldsymbol{y}} + \nu}_{=0})| \tag{14}$$

$$= \frac{1}{||\boldsymbol{\xi}||}|\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}^* + \nu| \tag{15}$$

$$= \frac{1}{||\boldsymbol{\xi}||}, \tag{16}$$

where we have used that $\tilde{\boldsymbol{y}} \in \mathcal{P}$ and $|\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}^* + \nu| = 1$. Hence, the optimization problem writes as follows

$$\begin{aligned} &\underset{\boldsymbol{\xi},\nu}{\text{argmax}} \;\; \frac{1}{||\boldsymbol{\xi}||}, \quad \boldsymbol{\xi} \in \mathbb{R}^{2n'}, \nu \in \mathbb{R} \\ &\text{subject to} \;\; \min_{i\in[1:N]} |\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}_i + \nu| = 1, \end{aligned} \tag{17}$$

or, equivalently,

$$\begin{aligned} &\underset{\boldsymbol{\xi},\nu}{\text{argmin}} \;\; \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi}, \quad \boldsymbol{\xi} \in \mathbb{R}^{2n'} \\ &\text{subject to} \;\; l_i(\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}_i + \nu) \geq 1, \quad \forall i \in [1:N], \end{aligned} \tag{18}$$

where we have used that $|\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}_i + \nu| = l_i(\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}_i + \nu)$. We then employ the Lagrangian formulation to include the constraint into the objective function as follows

$$\begin{aligned} &\underset{\boldsymbol{\xi},\nu,\boldsymbol{\alpha}}{\text{argmax}} \ \mathcal{L}(\boldsymbol{\xi},\nu,\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} - \sum_{i=1}^{N}\alpha_i(l_i(\boldsymbol{\xi}^T\tilde{\boldsymbol{y}}_i + \nu) - 1), \quad \boldsymbol{\xi},\boldsymbol{\alpha} \in \mathbb{R}^{2n'}, \nu \in \mathbb{R} \\ &\text{subject to} \quad \alpha_i \geq 0 \quad \forall i \in [1:N]. \end{aligned} \tag{19}$$

Next, we compute the gradient with respect to $\boldsymbol{\xi}$ and the derivative with respect to ν

$$\begin{cases} \nabla_{\boldsymbol{\xi}}\mathcal{L} = \boldsymbol{\xi} - \sum_{i=1}^{N}\alpha_i l_i \tilde{\boldsymbol{y}}_i = 0 \ \Rightarrow \ \boldsymbol{\xi} = \sum_{i=1}^{N}\alpha_i l_i \tilde{\boldsymbol{y}}_i \\ \dfrac{\partial \mathcal{L}}{\partial \nu} = -\sum_{i=1}^{N}\alpha_i l_i = 0. \end{cases} \tag{20}$$

Finally, substituting these expressions into (19), we get a formulation of the optimization problem that only depends on $\boldsymbol{\alpha}$

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{argmax}} \ \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} l_i l_j \alpha_i \alpha_j \tilde{\boldsymbol{y}}_i^T \tilde{\boldsymbol{y}}_j, \quad \boldsymbol{\alpha} \in \mathbb{R}^{2n'} \\ &\text{subject to} \quad \alpha_i \geq 0 \quad \forall i \in [1:N] \quad \text{and} \quad \sum_{i=1}^{N}\alpha_i l_i = 0. \end{aligned} \tag{21}$$

The problem (21) is solved using quadratic programming, yielding a solution $\boldsymbol{\alpha}^\star = (\alpha_1, \alpha_2, ..., \alpha_N)$. This vector is then used to compute the hyperplane given by

$$\boldsymbol{\xi}^\star = \sum_{i=1}^{N}\alpha_i^\star l_i \tilde{\boldsymbol{y}}_i, \tag{22}$$

and the following Karush–Kuhn–Tucker (KKT) given by

$$\alpha_i^\star(l_i(\boldsymbol{\xi}^{\star T}\tilde{\boldsymbol{y}}_i + \nu^\star) - 1) = 0, \quad \forall\, i \in [1:N] \tag{23}$$

are then exploited to yield the value of ν since, for all $j \in [1:N]$ such that $\alpha_j > 0$

$$l_i(\boldsymbol{\xi}^{\star T}\tilde{\boldsymbol{y}}_j + \nu^\star) = 1. \tag{24}$$

All the points that satisfy the condition (24) constitute the closest points to the hyperplane $\mathcal{P}$, and are called *support vectors*. We may solve (24) using any one of such points. Figure 3 shows a schematic representation of an SVM classifier. In sum, learning a classifier from the labeled dataset amounts to learning a *decision function* $f(\cdot)$ such that for all $\tilde{\boldsymbol{y}} \in \mathbb{R}^{2n'}$,

$$\begin{cases} f(\tilde{\boldsymbol{y}}) > 0 \implies \tilde{\boldsymbol{y}} \in \mathcal{C}_1 \\ f(\tilde{\boldsymbol{y}}) \leq 0 \implies \tilde{\boldsymbol{y}} \in \mathcal{C}_0. \end{cases} \tag{25}$$

3.2 SVM for linearly non-separable data

In many applications, and depending on the distribution of the dataset, we may be interested in producing a nonlinear classification function. In this scenario, we can resort to a technique called the *kernel method* whose basic idea consists in transforming the data

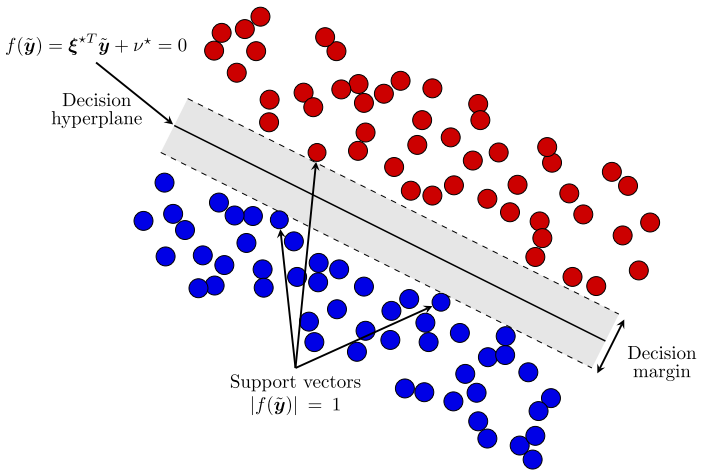

Fig. 3 Visual representation of an SVM classifier. The two classes (red and blue) are divided by a hyperplane that is at an equal and maximum distance from the nearest points in each class

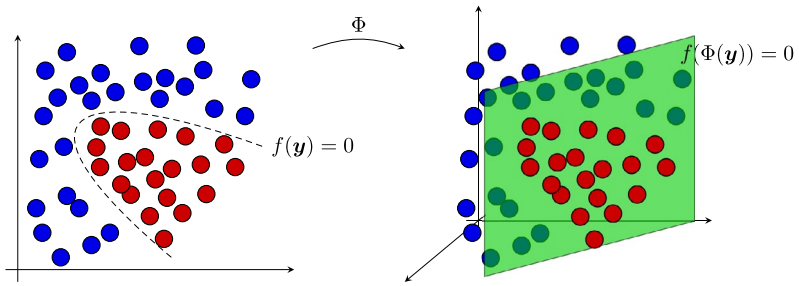

Fig. 4 Visual representation of the kernel method. In its original form, the data are not linearly separable. However, through the transformation Φ, the space is now three-dimensional and the data can be separated with a plane

into a high-dimensional space, where the data *become* linearly separable (see Figure 4). For instance, considering a dataset of vectors $\{\tilde{\boldsymbol{y}}_i\}_{1\leq i\leq N}$ of dimension 2, we could apply the following polynomial transformation

$$\phi(\tilde{\boldsymbol{y}}) = \phi(\tilde{y}_1, \tilde{y}_2) = (1,\, \tilde{y}_1,\, \tilde{y}_2,\, \tilde{y}_1\tilde{y}_2,\, \tilde{y}_1^2,\, \tilde{y}_2^2), \tag{26}$$

which would allow for a more complex—and thus powerful—separating function. However, observe that in the final optimization problem (21), the objective function does not depend explicitly on the data $\tilde{\boldsymbol{y}}_i$, but rather on the pairwise Euclidean inner product $\langle \tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_j \rangle = \tilde{\boldsymbol{y}}_i^T \tilde{\boldsymbol{y}}_j$, for $i, j \in [1 : N]^2$. Following a similar mathematical deduction to that of Section 3.1, it is easy to conclude that, if we project the data onto a high-dimensional space through a nonlinear function $\Phi(\cdot)$, the optimization problem (21) can be expressed as follows

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{argmax}}\ \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} l_i l_j \alpha_i \alpha_j K(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_j), \quad \boldsymbol{\alpha} \in \mathbb{R}^{2n'} \\ &\text{subject to} \quad \alpha_i \geq 0 \quad \forall i \in [1:N] \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i l_i = 0, \end{aligned} \tag{27}$$

where $K(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_j) = \langle \Phi(\tilde{\boldsymbol{y}}_i), \Phi(\tilde{\boldsymbol{y}}_j) \rangle$ denotes the inner product in the image space defined by $\{\Phi(\tilde{\boldsymbol{y}}) : \tilde{\boldsymbol{y}} \in \mathbb{R}^{2n'}\}$. As a consequence of this, we do not need to explicitly transform the data into the high-dimensional space, but only apply pairwise inner products. In this work, we will employ the well-known radial basis function (RBF), defined by

$$K(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{y}}') \triangleq e^{-\gamma ||\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{y}}'||^2}, \ \gamma \in \mathbb{R}^+, \tag{28}$$

which corresponds to the inner product over the infinite-dimensional space of exponentials.

Finally, let us observe that the absolute value $|f(\tilde{\boldsymbol{y}})|$ acts as a *reliability* measure: While $|f(\tilde{\boldsymbol{y}})| = 1$ indicates a support vector, $|f(\tilde{\boldsymbol{y}})| > 1$ indicates that the point is further away from the separating hyperplane and thus is more likely to belong to that class.

4 Related works and motivation

Previous works have already tackled the problem of SVM for channel coding [16–18, 20], where multiclass classification is needed. The two main strategies for multiclass SVM are the so-called *one vs. one* and *one vs. rest* approaches. The work in [20] employs SVMs for online adaptive modulation and coding, displaying favorable results with reduced complexity compared to previous algorithms. The authors in [16] proposed a novel pairwise SVM-based decoder that achieves competitive performances on convolutional codes, albeit at a very high complexity cost. The same analysis applies to the works in [17] and [18], which propose small modifications in training and application scenarios but keep the one vs. one classification approach. For a channel code of length n and dimension k, this technique has two main limitations:

(i) the dataset, which is composed of several noisy realizations of every valid codeword, grows exponentially with k (as the codeword space expands) and with n (as the number of correctable noise patterns increases);

(ii) the decoder contains a number of binary classifiers that also increases exponentially with k (due to the nature of the pairwise approach).

These two aspects result in an intractable decoder when applied to a code that is longer than a few bits, usually a Hamming code of size (7, 4). The largest implemented code is a BCH of size (15, 7) in [18] and employs up to 100 noisy realizations of each valid codeword.

We will now describe the one vs. rest and one vs. one approaches, along with their limitations and complexity constraints. In what follows, we will assume that the training dataset $\{\tilde{\boldsymbol{y}}_i, l_i\}_{1 \leq i \leq N}$ contains several noisy realizations of every valid codeword, where

- $\tilde{\boldsymbol{y}}_i \in \mathbb{R}^{2n'}$ denotes a received (complex) signal with its real and imaginary parts concatenated, corresponding to a transmitted message $\boldsymbol{u}_i$ as in Figure 1;

- l_i indicates the ground truth class to which $\tilde{\boldsymbol{y}}_i$ belongs, i.e., the transmitted message $\boldsymbol{u}_i$. A label l_i indicates belonging to the class $\mathcal{C}_i$, $i \in [1 : 2^k]$, which corresponds to a possible message (for instance, the binary representation of i using k bits).

4.1 One vs. rest

The *one vs. rest* method is based on producing 2^k binary SVM decision functions $f^{(j)}$ for $j \in [1 : 2^k]$, each one isolating one class (i.e., one codeword) against all the others [21]. This leaves a binary layout where the value $f^{(j)}(\tilde{\boldsymbol{y}})$ indicates whether the element $\tilde{\boldsymbol{y}}$ belongs to the class $\mathcal{C}_j$. We repeat this process for every class and end up with 2^k SVM classifiers, each corresponding to a valid codeword. All the SVMs are then applied to the received signal $\tilde{\boldsymbol{y}}$, and the selected class $\mathcal{C}_{j^\star}$ (codeword) is such that

$$j^\star = \underset{j \in [1:2^k]}{\operatorname{argmax}} f^{(j)}(\tilde{\boldsymbol{y}}). \tag{29}$$

This corresponds to the class that has the largest distance between the point $\tilde{\boldsymbol{y}}$ and the separating hyperplane. If no decision function $f^{(j)}(\tilde{\boldsymbol{y}})$ gives a positive outcome, then the nearest class is selected, i.e., $f^{(j)}(\tilde{\boldsymbol{y}})$ with the negative value closest to 0 (Fig. 5).

4.2 One vs. one

The one vs. one approach is employed in [16–18], and it consists in producing pairwise SVM classifiers for all possible combinations of valid codewords, yielding a set of functions $f^{(j,j')}$ for all $(j, j') \in [1 : 2^k] \times [1 : 2^k]$. The total number of SVM functions can be easily computed

$$\binom{2^k}{2} = \frac{2^k!}{2!(2^k-2)!} = \frac{2^k(2^k-1)(2^k-2)!}{2(2^k-2)!} = 2^{k-1}(2^k-1). \tag{30}$$

Once we have produced the $2^{k-1}(2^k - 1)$ SVM binary classifiers, every function is applied to the received signal $\tilde{\boldsymbol{y}}$, and the selected class is determined through a voting system, given by

$$j^\star = \underset{j \in [1:2^k]}{\operatorname{argmax}} \sum_{\substack{j'=1 \\ j' \neq j}}^{2^k} \mathbb{1}\left(f^{(j,j')}(\tilde{\boldsymbol{y}}) > 0\right). \tag{31}$$

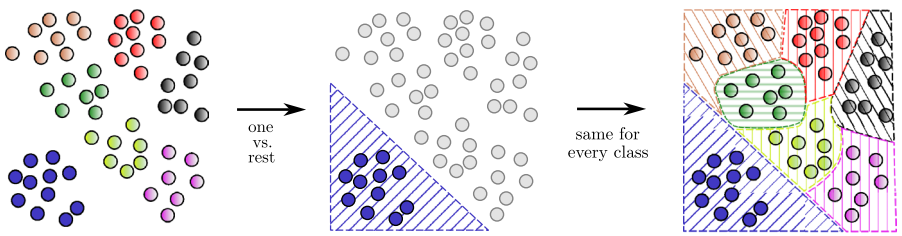

Fig. 5 Visual representation of the one vs. rest approach. Each class is put against all others to produce a binary classifier. The procedure is repeated 2^k times, once for each valid codeword

Each decision function decides between two classes (codewords), and the class that gets the most votes at the end is selected. Ties can be resolved either randomly or by considering the mean of the votes' reliabilities, i.e., the distance to the separating hyperplanes.

5 Proposed system and training framework

In the following, we describe our suggested solution, which combines a bit-wise approach to the SVM decoder with a noiseless optimization procedure [22]. Finally, a theoretical analysis is given that studies the relationship between our proposed solution and the bit-ML decoder.

5.1 Bit-wise SVM

The previous approaches present the main constraint of a complexity that increases exponentially with the message length, with at least 2^k SVMs for a code of size (n, k)—and even more for the one vs. one approach. To alleviate this constraint, we suggest a novel bit-wise approach that resorts to only k SVM classifiers for an (n, k) linear block code. This method transforms the multiclass problem generating one SVM per valid codeword, into a series of k binary classifications necessitating one SVM per bit position, as shown in Figure 6. To this end, for all $j \in [1 : k]$, we divide the dataset $\{\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_N\}$ into two non-intersecting classes

- $\mathcal{C}_1^{(j)}$ corresponding to the vectors for which the transmitted message $\boldsymbol{u} = (u_1, ..., u_k)$ satisfies $u_j = 1$;
- $\mathcal{C}_0^{(j)}$ corresponding to the vectors for which $u_j = 0$.

For each $j \in [1 : k]$, an SVM classifier $f^{(j)}$ is produced such that if $f^{(j)}(\tilde{\boldsymbol{y}}) > 0$, then $\hat{u}_j = 1$, and $\hat{u}_j = 0$ otherwise. Consequently, each decision function $f^{(j)}$, for $j \in [1 : k]$, will decide whether the jth bit of the estimated message $\hat{\boldsymbol{u}}$ is a 0 or a 1. Algorithm 1 outlines the decoding iterative process. The suggested bit-wise SVM not only reduces the number of SVMs necessary from 2^k to k, but can be implemented in parallel in order to reduce latency.

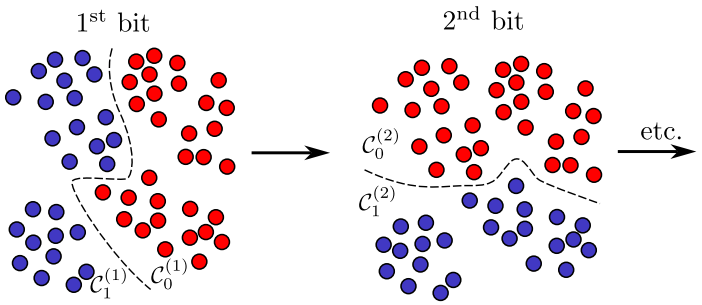

Fig. 6 Visual representation of the proposed bit-wise approach

Input: $\boldsymbol{y}$ a noisy codeword of size n
Output: $\hat{\boldsymbol{u}}$ an estimated binary message of size k
Initialization: $f^{(j)}$ SVM bit classifier for $j \in [1:k], \hat{\boldsymbol{u}} = \vec{0}_k$
$\tilde{\boldsymbol{y}} = [\mathcal{R}(\boldsymbol{y}), \mathcal{I}(\boldsymbol{y})]$
for $j = [1:k]$ **do**
  if $f^{(j)}(\tilde{\boldsymbol{y}}) \geq 0$ **then**
    $\hat{u}_j = 1$
  else
    $\hat{u}_j = 0$
  end if
end for
return $\hat{\boldsymbol{u}}$

5.2 Noiseless optimization framework

To further reduce the complexity with respect to the SVM-based solutions in [16–18], we make use of a particularly appealing feature of SVMs, namely their maximum margin property, which yields separating hyperplanes that are equidistant from both dataset classes. As such, when investigating symmetric channel models like the AWGN, this is equivalent to a maximum margin classifier between *only* the original noiseless codewords (i.e., the classes' centroids). Consequently, rather than the traditional training approach which considers a dataset with randomly generated noisy codewords, it suffices to optimize—or *train*—the suggested bit-wise SVM on only noiseless modulated codewords as shown in Figure 7.

[Style1 Style2]Observation 1 Training on only one sample per class—i.e., the noiseless codeword—is possible due to the specific maximum margin property of SVM. This would not be a straightforward application in regular neural network-based solutions, which would likely overfit their decision regions to noiseless transmission and generalize poorly to noisier conditions as shown in [23] and analyzed in [24].

The suggested noiseless optimization not only drastically reduces the size of the training dataset but also allows to be robust to possible mismatches between the training and actual channel conditions (e.g., different training and testing SNR). The new training dataset contains only one sample per class, i.e., 2^k elements, which

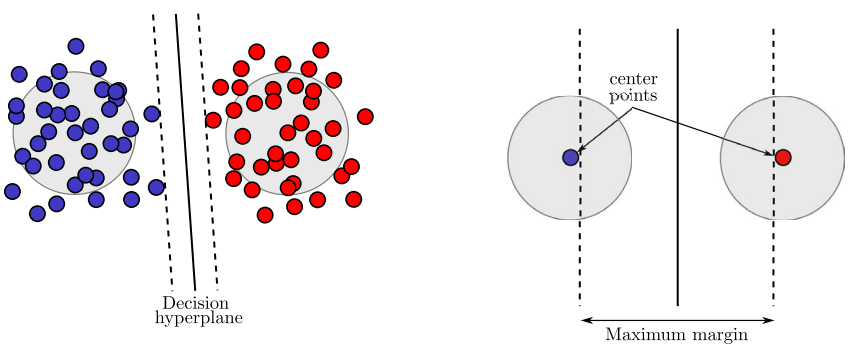

Fig. 7 Visual demonstration of noiseless optimization: noisy scenario (left) and noiseless scenario (right)

correspond to the noiseless modulated codewords $\{\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_{2^k}\}$, where the same processing of (1) is applied to every valid modulated codeword.

5.3 Proposed optimization problem

Combining the two suggested elements (bit-wise SVM and noiseless optimization), the training dataset will be composed of the 2^k valid modulated codewords $\{\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_{2^k}\}$ with $2n'$ elements each—where the same preprocessing (1) has been applied to $\boldsymbol{x}$—and k binary classifiers will be produced following the bit-wise approach of Section 5.1. Because we are working with linearly non-separable spaces, the kernel method is employed, and the RBF of (28) is selected as the kernel function. The classification of an unlabeled vector $\tilde{\boldsymbol{y}}$ consists of k classifiers $f^{(j)}(\tilde{\boldsymbol{y}})$, each determining the value of the jth bit of the estimated message $\hat{\boldsymbol{u}}$, and given by

$$f^{(j)}(\tilde{\boldsymbol{y}}) = \sum_{i=1}^{2^k} l_i^{(j)} \alpha_i^{(j)} K(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}) + \nu^{(j)}, \tag{32}$$

where $l_i^{(j)} = +1$ if the jth information bit of the modulated codeword $\boldsymbol{x}_i$ is 1, and $l_i^{(j)} = -1$ otherwise. Lastly, $\boldsymbol{\alpha}^{(j)}$ constitutes the solution to the following optimization problem

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \ \frac{1}{2} \sum_{i,m=1}^{2^k} \alpha_i \alpha_m l_i^{(j)} l_m^{(j)} K(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_m) - \sum_{i=1}^{2^k} \alpha_i \\ &\quad \text{subject to: } \alpha_i \geqslant 0 \text{ and } \textstyle\sum_{i=1}^{2^k} l_i^{(j)} \alpha_i = 0. \end{aligned} \tag{33}$$

Each of the k bit-wise SVM optimization problems given in (33) can be written as a quadratic programming problem with linear constraints given by

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \ \frac{1}{2} \boldsymbol{\alpha}^T Q^{(j)} \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha} \\ &\text{subject to: } \boldsymbol{\alpha} \geq \mathbf{0} \text{ and } \boldsymbol{\alpha}^T \boldsymbol{l}^{(j)} = 0, \end{aligned} \tag{34}$$

where $Q^{(j)}$ is a matrix such that $Q_{i,m}^{(j)} = l_i^{(j)} l_m^{(j)} K(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_m)$. Since all $Q^{(j)}$ are definite positive matrices, these optimization problems are all convex.

5.4 Optimality analysis and interpretation

Although obtaining closed-form solutions of (33) for generic choices of the (n, k) linear block code, the constellation, and the parameter γ might be challenging, we show that under certain assumptions, the resulting SVM decision rule can be obtained in closed-form and related to the bit-ML decision rule, which, in turn is equivalent to the bit-MAP rule under i.i.d. input bits assumption.

Theorem 1 (Optimal solution and equivalence to bit-ML)

(i) *For $\gamma \gg 1$, the optimal solution to (33) for all $j \in [1 : k]$ is given by $\boldsymbol{\alpha}^* = (1, 1, ..., 1)$, and $\nu^* = 0$.*

(ii) *Furthermore, if* $\gamma = 1/\sigma^2$, *this solution yields decision functions* $f^{(j)}(\tilde{\boldsymbol{y}})$ *equal to the bit-ML decision rule* $g^{(j)}(\tilde{\boldsymbol{y}})$ *of* (4)

1 *Proof*

In order to prove (i), let us notice that if $\gamma \gg 1$, *then one can show that for all* $i \neq m$, $K(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_m) \approx 0$. *Hence, the objective function (34) can be expressed, for all* $j \in [1:k]$, *as*

$$\frac{1}{2}\sum_{i=1}^{2^k}\alpha_i^2 - \sum_{i=1}^{2^k}\alpha_i = \frac{1}{2}\sum_{i=1}^{2^k}(\alpha_i^2 - 1)^2 - 2^{k-1}. \tag{35}$$

One can then easily show that the solution $\boldsymbol{\alpha}^* = (1, 1, ..., 1)$ yields a lower bound to the objective function, since $\sum_{i=1}^{2^k}(\alpha_i^2 - 1)^2 \geq 0$, and verifies the inequality constraints $\alpha_i \geq 0\ \forall i \in [1:2^k]$. The equality constraint is also verified, since for all $j \in [1:k]$, each of the two classes $\mathcal{C}_1^{(j)}$ and $\mathcal{C}_0^{(j)}$ consist of 2^{k-1} sequences $\tilde{\boldsymbol{x}}_i$ and hence,

$$\sum_{i=1}^{2^k} l_i^{(j)} = |\mathcal{C}_0^{(j)}| - |\mathcal{C}_1^{(j)}| = 0 \quad \forall j \in [1:k]. \tag{36}$$

Thus, for all $j \in [1:k]$, $\boldsymbol{\alpha}^* = (1, 1, ..., 1)$ is the solution to the optimization problem in (33). As for $\nu^{(j)}$, note that by evaluating (32) for any $\tilde{\boldsymbol{x}}_m$ using $\gamma \gg 1$, we obtain

$$f^{(j)}(\tilde{\boldsymbol{x}}_m) = \sum_{i=1}^{2^k} l_i^{(j)} e^{-\gamma||\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_m||^2} + \nu^{(j)} = l_m^{(j)} + \nu^{(j)}, \tag{37}$$

which yields $\nu^{(j)} = 0$.

To prove (ii), let us observe that replacing the proposed solution $\{\boldsymbol{\alpha}^* = (1, 1, ..., 1), \nu^* = 0\}$ in (32) yields

$$f^{(j)}(\tilde{\boldsymbol{y}}) = \sum_{i=1}^{2^k} l_i^{(j)} e^{-\gamma||\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}||^2}, \tag{38}$$

where $l_i^{(j)}$ denotes the label of the point $\tilde{\boldsymbol{x}}_i$ for the jth classification problem (related to the value of its jth bit). Let us divide the summation argument into the two classes $\mathcal{C}_1^{(j)}$ and $\mathcal{C}_0^{(j)}$. Hence, we can rewrite the decision function as

$$f^{(j)}(\tilde{\boldsymbol{y}}) = \sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{C}_1^{(j)}} e^{-\gamma||\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}||^2} - \sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{C}_0^{(j)}} e^{-\gamma||\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}||^2}. \tag{39}$$

From (39), it is easy to deduce the value of the jth bit as

$$g^{(j)}(\tilde{\boldsymbol{y}}) = \mathbb{1}\left\{ \sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{C}_1^{(j)}} e^{-\gamma||\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}||^2} > \sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{C}_0^{(j)}} e^{-\gamma||\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}||^2} \right\}. \tag{40}$$

Selecting $\gamma = 1/\sigma^2$ in (40) yields the bit-ML rule in (4) for an AWGN channel with noise power σ^2 by noticing that

$$P_{\boldsymbol{Y}|X}(\boldsymbol{y}|\boldsymbol{x}(\boldsymbol{u})) = \frac{1}{(\pi\sigma^2)^{n'}} e^{-\frac{||\boldsymbol{x}(\boldsymbol{u})-\boldsymbol{y}||^2}{\sigma^2}}. \tag{41}$$

□

In the following, we will show that the assumption of $\gamma \gg 1$ in Theorem 1 is valid even for moderate SNR values. Besides, we will analyze the effect of relaxing the constraint $\gamma = 1/\sigma^2$ on the obtained results with respect to the bit-ML.

6 Results and discussion

This section will present the simulation results obtained using the proposed approach, combining the bit-wise decoding framework and the noiseless optimization. The system model of Section 5.1 was employed along with a simple Monte Carlo simulation to compute the bit error rate (BER), defined by

$$\text{BER} \triangleq \frac{1}{N'} \sum_{i=1}^{N'} \left(\frac{1}{K} \sum_{j=1}^{k} \mathbb{1}(g_j(\mathbf{y}^i) \neq u_j^i) \right) \tag{42}$$

where N' is the number of Monte Carlo samples $(\mathbf{u}^i, \mathbf{y}^i)$, which is adapted to meet a stopping criterion of 1000 frame errors for each considered E_b/N_0. Optimization problems are solved using the CVX solver for convex optimization [25, 26]. The parity-check matrices were taken from the channel code database in [27]. Observe that previous solutions (i.e., one vs. one and one vs. rest) were not implemented due to their extremely high complexity, even for the short codes simulated. For results on shorter codes, the reader is referred to [16, 18].

6.1 BER and effect of the hyperparameter γ

The suggested bit-wise SVM was implemented following the communication chain of Figure 1 for both an extended (32, 11) BCH code and a (32, 11) Polar code, each under a 16-QAM modulation scheme and an AWGN channel with an $E_b/N_0 = \frac{n}{k.m}\frac{1}{\sigma^2}$, where $m = 4$ denotes the order of the modulation. Random messages of length k are generated in each iteration, coded to n bits, modulated, and transmitted through the AWGN channel. Let us define γ_s the value of the exponential's slope

$$\gamma_s = 1/\sigma_s^2, \tag{43}$$

where σ_s^2 is the noise power such that $E_b/N_0 = s$dB. This is what is referred to as a value of γ *adapted* to a normalized signal-to-noise ratio of $E_b/N_0 = s$dB. The proposed system is trained with different values of γ. Figures 8 and 9 display the BER performances of the proposed solution trained with γ_0, γ_5 and γ_{10} (adapted to 0dB, 5dB, and 10dB, respectively). Therein, we confirm the result of Theorem 1, where a larger value of γ showcases a BER that coincides with the bit-ML decoding rule, whereas a lower value presents a

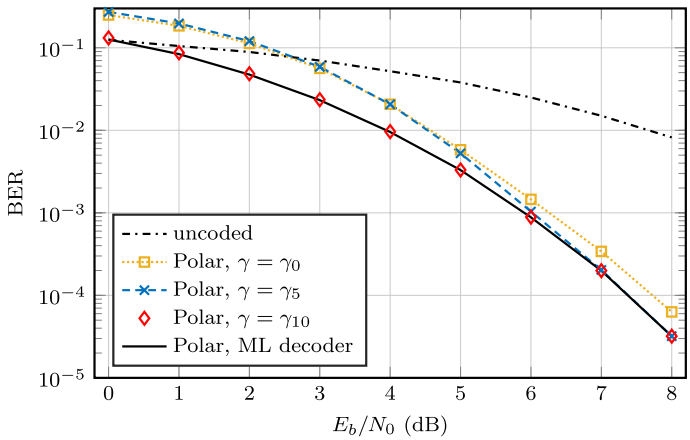

Fig. 8 BER of the suggested decoder for both a Polar code of size (32, 11), with a 16-QAM modulation and under an AWGN channel

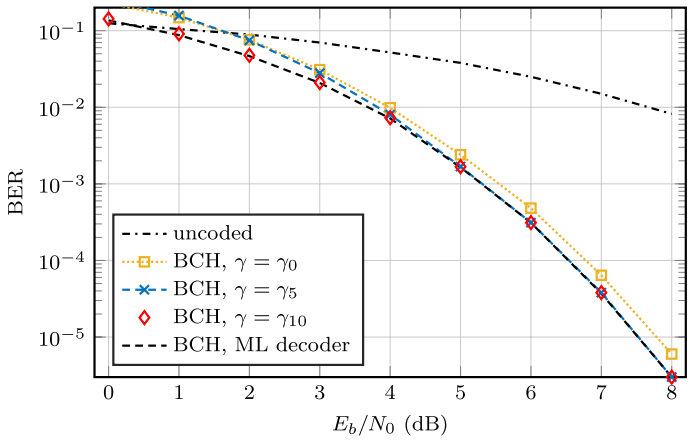

Fig. 9 BER of the suggested decoder for both a BCH code of size (32, 11), with a 16-QAM modulation and under an AWGN channel

performance gap with respect to optimal decoding. The same applies to both the Polar and BCH codes.

To assess the effect of the choice of the parameter γ, Figure 10 shows the E_b/N_0 corresponding to a BER of 10^{-3} as a function of $s \in [-2 : 12]$ for both the (32, 11) BCH code and the (32, 11) Polar code. One can observe that very low values of s—and thus their corresponding values of γ_s—display poorer performances in terms of BER. However, above a threshold value of around 2dB, the SVM achieves the objective BER of 10^{-3} at essentially the same E_b/N_0 as the bit-ML decoding solution. This phenomenon suggests that training with large values of γ relaxes the need to adapt γ to the current E_b/N_0, generalizing in this way the result of Theorem 1, (ii).

Moreover, as previously discussed, the result of Theorem 1, (i) is valid for γ corresponding to even moderate values of E_b/N_0. Figure 11 shows the solution to the optimization problem (33) as a function of $s \in [-2 : 12]$. We observe that, indeed, the optimal solution to (33) is given by $\boldsymbol{\alpha}^* = (1, 1, ..., 1)$ and $\nu^* = 0$ for all $s >$ 2dB, which corresponds to relatively moderate values of E_b/N_0.

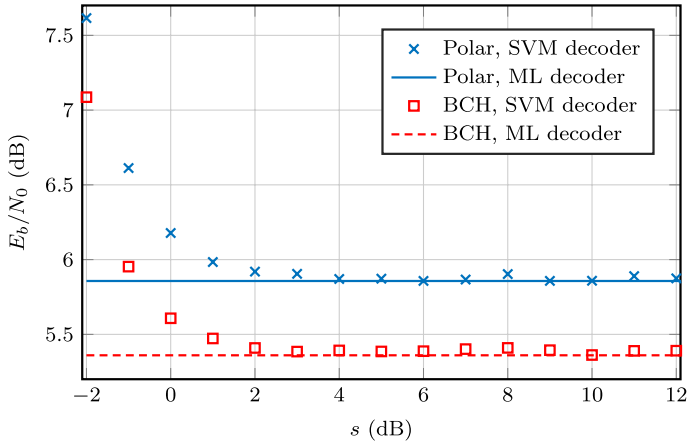

Fig. 10 E_b/N_0 corresponding to a BER of 10^{-3} as a function of s

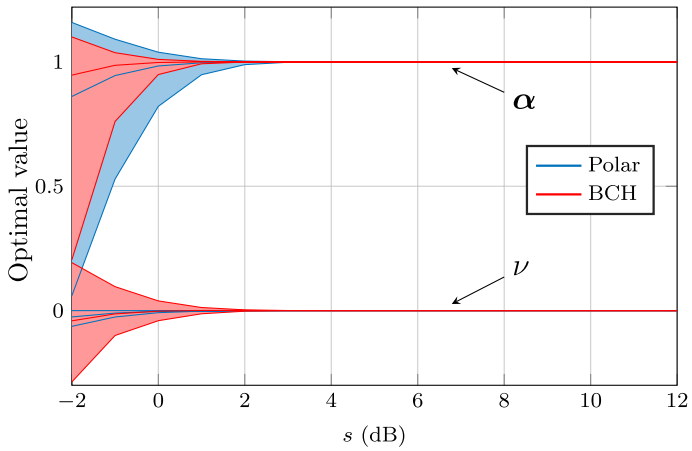

Fig. 11 Optimal values of $\boldsymbol{\alpha}$ and ν—i.e., solutions to the optimization problem (33)—as a function of s

Table 1 Complexity comparison between methods

	Bit-wise	One vs. rest	One vs. one
# of SVM classifiers	k	2^k	$2^{k-1}(2^k-1)$
# training samples	$N=2^k$	$N=L.2^k$	$N=L.2^k$
maximum # of terms in (32)	$N=2^k$	$N=L.2^k$	$\approx \frac{N}{2^{k-1}}=2.L$
maximum # of terms in (32) with noiseless opt.	2^k	2^k	2

6.2 Complexity analysis

Table 1 summarizes the training and decoding complexity of our method against other existing SVM-based methods in the literature, i.e., the one vs. rest method of [21] and the one vs. one method of [16–18]. As we can observe, our proposed bit-wise approach improves over existing methods since it enables a linearly growing number of SVMs instead of the exponential number of SVMs in the existing solutions, which

allows for better scalability. Second, using the noiseless optimization idea, our training dataset size was reduced to its minimum $N = 2^k$, i.e., it suffices to train the SVM using the centroids of the different classes. This contrasts with existing methods that use noisy datasets to train their SVMs, thus incurring a dataset size of $N = L.2^k$, with $L \in \mathbb{N}^+$ often taken as $L = 100$ in the literature.

Besides, to assess the overall decoding complexity, we consider not only the number of SVM classifiers but also the number of computations required to perform each one of these SVM classifications. The size of the training dataset is equal to 2^k; if all possible codewords are support vectors such as for the AWGN case with RBF kernels, i.e., $\alpha_i = 1$ for all $i \in [1 : 2^k]$, then the classification task of our solution (32) requires the computation of a maximum of 2^k terms. The one vs. one method and one vs. rest can also be adapted using a noiseless optimization, in which case, the one vs. one method provides the least number of operations (2 operations per classification). However, the one vs. one method requires an exponential number of SVM classifiers with respect to our solution.

To conclude, our method provides for the lowest overall decoding complexity when considering both the number of SVM classifiers and the number of computations per classifier. Hence, while existing contributions are restricted to very short codes, Hamming (7, 4) or BCH (15, 7), for instance, our proposed solution allows for larger message lengths. However, although reduced with respect to the existing solutions, the size of the training dataset, and hence the maximum number of support vectors, remains exponential in k, which renders our SVM-based decoding intractable for values of $k > 18$ bits.

6.3 Effect of the Kernel function

As we have discussed in Section 6.2, the main complexity of our solution arises from the number of terms in (32), which is exponential in k. This is due to the fact that the RBF kernel yields an optimization function where, in its solution, every element in the dataset constitutes a support vector (see Theorem 1).

Various classical kernels (see [28]) were implemented in order to evaluate the resulting performance. The BER for the polynomial kernel—defined as $K_{\text{poly}}(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{y}}') \triangleq (1 + \tilde{\boldsymbol{y}}^T \tilde{\boldsymbol{y}})^d$, with $d \in \mathbb{N}^+$ is shown in Figure 12 for both considered codes. Even though the obtained BER is better than random guessing (BER = 0.5), the performances suffer from a significant reduction compared to the RBF kernel (i.e., bit-ML decoding) of Figure 9 and Figure 8. The other considered kernels present similar behaviors, where support vectors are only a subset of all possible codewords, but the BER performance is greatly reduced.

6.4 SVM vs. separate demodulation and decoding

In the following, we emphasize the benefit of using SVM for joint demodulation and decoding as compared to using separate demodulation and decoding schemes, which is classical in digital communication systems. The separate demodulation and decoding architecture consists in, first, a soft-MAP demodulator which maps each of the n' noisy received symbols $\boldsymbol{y}$ into the log-likelihood ratios (LLRs) of the n coded bits $\boldsymbol{c}$, defined for all $j \in [1 : n]$ as

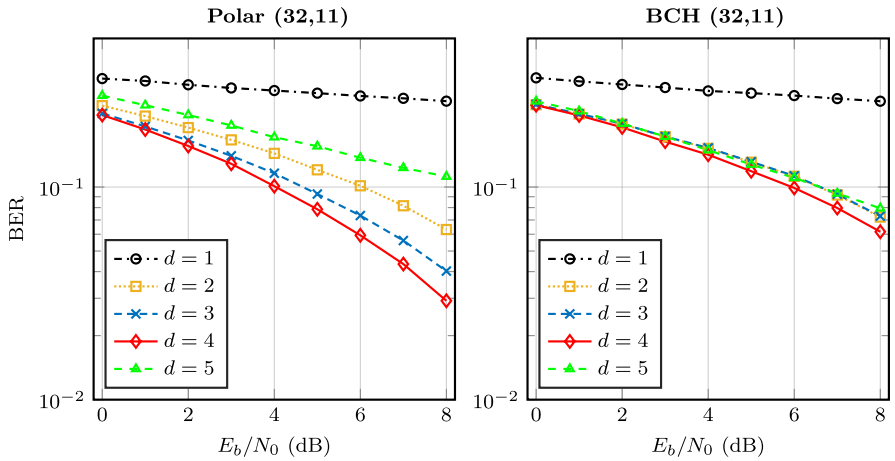

Fig. 12 BER for both a Polar (**left**) and BCH (**right**) codes of size (32, 11), with a 16-QAM modulation and under an AWGN channel, employing the polynomial kernel of degree *d*

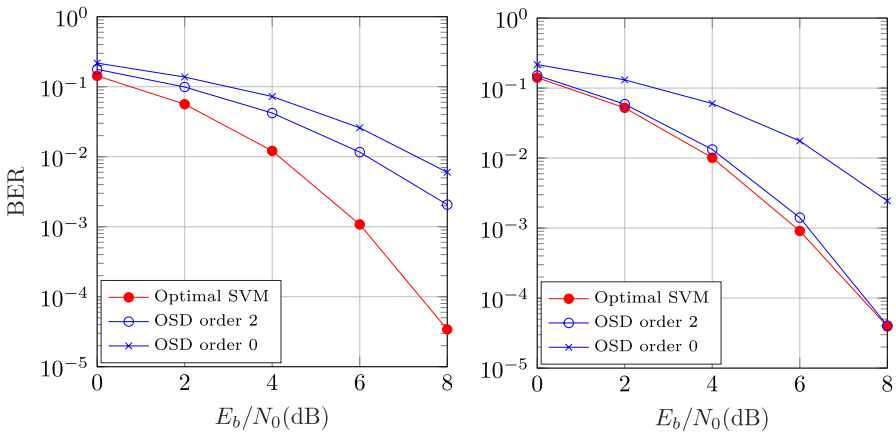

Fig. 13 BER of a (32, 11) Polar code with 16-QAM under both *binary* mapping (**left**) and *Gray* mapping (**right**), decoded by an optimal SVM or by an OSD

$$l_j(y_{\lceil j/m \rceil}) \triangleq \log\left(\frac{P_{Y_{\lceil j/m \rceil}|C_j}(y_{\lceil j/m \rceil}|0)}{P_{Y_{\lceil j/m \rceil}|C_j}(y_{\lceil j/m \rceil}|1)}\right). \tag{44}$$

Then, the obtained LLRs are fed to a quasi-optimal decoder, namely the ordered statistics decoder (OSD) [29] whose order is chosen high enough to ensure the convergence to an ML-decoder but given the LLRs at its inputs (and not the received symbols as in the decoder of (3)).

Separate demodulation and decoding is known to be sufficient for constellations under *Gray* mapping; however, it is sub-optimal for other symbol mappings such as the *binary* mapping [30]. This is due to the fact that the demodulation is performed symbol-wise and does not fully exploit possible correlation between coded bits of different symbols.

Figure 13 shows the performance of the resulting separate demodulation and OSD decoding with respect to the optimal SVM decoder, for a (32, 11) Polar code with 16-QAM and an AWGN channel model. We can clearly see that the OSD-based decoder

achieves (as the order grows) the same performance as the SVM for Gray-mapped constellations, confirming thus the optimality of both joint and separate demodulation and decoding. However, for the binary-mapped constellation, although an OSD decoder of order 2 approaches the ML decoder when given the LLRs as inputs, and not the noisy symbols $\boldsymbol{y}$, the overall decoder performance remains far from that of joint demodulation and decoding bit-ML detector. This is solved in practice by resorting to iterative decoding, such as in Bit-Interleaved Coded Modulation (BICM) [30].

7 Conclusion and perspectives

In this work, we have discussed the application of SVMs to the problem of channel decoding. After providing the reader with an introduction to SVMs and a state-of-the-art study on their applications to channel decoding, we have presented our proposed solution and training framework. This novel decoding system reduces complexity compared to the previous SVM-based methods: (i) The number of decision functions is reduced from (at least) 2^k to k; (ii) the number of elements in the dataset is reduced to a minimum of one element per possible codeword, i.e., 2^k; and (iii) in symmetric channels, robustness to the dataset's distribution is improved by only training on the noiseless codewords. Also, the proposed decoder is proven to be equivalent to bit-ML decoding for the AWGN channel under RBF kernels. Hence, the complexity involved in our solution remains intractable for longer codes. Besides, we showed that using SVM for joint demodulation and decoding allows to outperform the classical separate approach for some specified symbol mappings. This closes a door on the *SVM decoder for AWGN* debate, but also opens interesting research perspectives:

- Exploiting the code's structure in order to reduce the number of support vectors in the solution of the SVM optimization problem.
- Apply a similar approach to channel models for which the ML receiver cannot be obtained in closed form. Such is the case of impulsive noise in power line communications, which is modeled as an alpha-stable distribution, renowned for their lack of closed-form pdf [31].

These and many other questions may be addressed in future works, potentially combining other machine learning-based techniques to, for instance, learn smarter feature representations for the codeword space that reduce the complexity of the final SVM decision function.

List of abbreviations

AWGN	Additive white Gaussian noise
BCH	Bose–Chaudhuri–Hocquenghem
BEP	Bit error probability
BER	Bit error rate
FEC	Forward error correction
KKT	Karush–Kuhn–Tucker
MAP	Maximum a posteriori
ML	Maximum likelihood
pdf	Probability density function
PSK	Phase-shift keying
QAM	Quadrature amplitude modulation
RBF	Radial basis function
SNR	Signal-to-noise ratio

SVM Support vector machine

Author contributions
GDBR and MB proposed the main contributions and provided the mathematical proofs. HM and TB provided insights on channel decoding constraints and feasibility. All authors read and approved the final manuscript.

Funding
Not applicable.

Data availability
All the data were obtained via computer simulations, and the error correction codes employed were taken from the channel code database of [27]. The code to reproduce the experiments can be found in [32].

Declarations

Competing interests
Not applicable.

Received: 22 October 2024 Accepted: 8 August 2025
Published online: 27 August 2025

References

1. M.A. Alsheikh, S. Lin, D. Niyato, H.-P. Tan, Machine learning in wireless sensor networks: algorithms, strategies, and applications. IEEE Commun. Surv. Tutor. **16**(4), 1996–2018 (2014). https://doi.org/10.1109/COMST.2014.2320099
2. C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, L. Hanzo, Machine learning paradigms for next-generation wireless networks. IEEE Wirel. Commun. **24**(2), 98–105 (2017). https://doi.org/10.1109/MWC.2016.1500356WC
3. C. Zhang, P. Patras, H. Haddadi, Deep learning in mobile and wireless networking: a survey. IEEE Commun. Surv. Tutor. **21**(3), 2224–2287 (2019). https://doi.org/10.1109/comst.2019.2904897
4. C. She, R. Dong, Z. Gu, Z. Hou, Y. Li, W. Hardjawana, C. Yang, L. Song, B. Vucetic, Deep learning for ultra-reliable and low-latency communications in 6G networks. IEEE Netw. **34**(5), 219–225 (2020). https://doi.org/10.1109/mnet.011.1900630
5. G. Zeng, D. Hush, N. Ahmed, An application of neural net in decoding error-correcting codes. In: 1989 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 782–7852 (1989). https://doi.org/10.1109/ISCAS.1989.100467
6. J. Bruck, M. Blaum, Neural networks, error-correcting codes, and polynomials over the binary n-cube. IEEE Trans. Inf. Theory **35**(5), 976–987 (1989). https://doi.org/10.1109/18.42215
7. J. Yuan, V.K. Bhargava, Q. Wang, An Error Correcting Neural Network. In: Conference Proceeding IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 530–533 (1989). https://doi.org/10.1109/PACRIM.1989.48418
8. T. Gruber, S. Cammerer, J. Hoydis, S.t. Brink, On deep learning-based channel decoding. In: 2017 51st Annual Conference on Information Sciences and Systems (CISS), pp. 1–6 (2017). https://doi.org/10.1109/CISS.2017.7926071
9. E. Nachmani, Y. Be'ery, D. Burshtein, Learning to decode linear codes using deep learning. In: 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 341–346 (2016). https://doi.org/10.1109/ALLERTON.2016.7852251
10. A. Bennatan, Y. Choukroun, P. Kisilev, Deep learning for decoding of linear codes—a syndrome-based approach. In: 2018 IEEE International Symposium on Information Theory (ISIT), pp. 1595–1599 (2018). https://doi.org/10.1109/ISIT.2018.8437530
11. Y. Choukroun, L. Wolf, Error Correction Code Transformer. arXiv (2022). https://doi.org/10.48550/ARXIV.2203.14966
12. B.E. Boser, I.M. Guyon, V.N. Vapnik, A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory. COLT92, pp. 144–152 (1992). https://doi.org/10.1145/130385.130401
13. C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1007/bf00994018
14. V.N. Vapnik, The Support Vector method, pp. 261–271 (1997). https://doi.org/10.1007/bfb0020166
15. M.J.F.-G. Garcia, J.L. Rojo-Alvarez, F. Alonso-Atienza, M. Martinez-Ramon, Support vector machines for robust channel estimation in OFDM. IEEE Signal Process. Lett. **13**(7), 397–400 (2006). https://doi.org/10.1109/lsp.2006.871862
16. J.W.H. Kao, S.M. Berber, Error control coding based on support vector machine. 2008 IAPR Workshop On Cognitive Information Processing, pp. 182–187 (2008)
17. R. Ramanathan, K.P. Soman, N. Valliappan, S.P. Mathavan, M. Gayathri, R. Priya, Generalised and channel independent SVM based robust decoders for wireless applications. In: 2009 International Conference on Advances in Recent Technologies in Communication and Computing (2009). https://doi.org/10.1109/artcom.2009.92
18. V. Sudharsan, B. Yamuna, Support vector machine based decoding algorithm for BCH codes. J. Telecommun. Inf. Technol. **2**(2016), 108–112 (2016). https://doi.org/10.26636/jtit.2016.2.728
19. J. Muramatsu, S. Miyake, On the error probability of stochastic decision and stochastic decoding. In: 2017 IEEE International Symposium on Information Theory (ISIT) (2017). https://doi.org/10.1109/isit.2017.8006808
20. R. Daniels, R.W. Heath, Online adaptive modulation and coding with support vector machines. In: 2010 European Wireless Conference (EW) (2010). https://doi.org/10.1109/ew.2010.5483527
21. C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines. IEEE Trans. Neural Netw. **13**(2), 415–425 (2002). https://doi.org/10.1109/72.991427

22. G. De Boni Rovella, M. Benammar, H. Méric, T. Benaddi, on the optimality of support vector machines for channel decoding. In: 2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit) (2024). https://doi.org/10.1109/eucnc/6gsummit60053.2024.10597010
23. T. O'Shea, J. Hoydis, An introduction to deep learning for the physical layer. IEEE Trans. Cognit. Commun. Netw. **3**(4), 563–575 (2017). https://doi.org/10.1109/tccn.2017.2758370
24. M. Benammar, P. Piantanida, Optimal training channel statistics for neural-based decoders. In: 2018 52nd Asilomar Conference on Signals, Systems, and Computers, pp. 2157–2161 (2018). https://doi.org/10.1109/ACSSC.2018.8645128
25. M. Grant, S. Boyd, Graph implementations for nonsmooth convex programs. In: Blondel, V., Boyd, S., Kimura, H. (eds.) Recent advances in learning and control. Lecture Notes in Control and Information Sciences, pp. 95–110 (2008). http://stanford.edu/~boyd/graph_dcp.html
26. M. Grant, S. Boyd, CVX: Matlab Software for Disciplined Convex Programming, version 2.1. https://cvxr.com/cvx (2014)
27. M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, N. Wehn, Database of Channel Codes and ML Simulation Results. www.uni-kl.de/channel-codes (2019)
28. A. Patle, D.S. Chouhan, SVM kernel functions for classification. In: 2013 International Conference on Advances in Technology and Engineering (ICATE), pp. 1–9 (2013). https://doi.org/10.1109/ICAdTE.2013.6524743
29. M.P.C. Fossorier, S. Lin, Soft-decision decoding of linear block codes based on ordered statistics. IEEE Trans. Inf. Theory **41**(5), 1379–1396 (1995). https://doi.org/10.1109/18.412683
30. G. Caire, G. Taricco, E. Biglieri, Bit-interleaved coded modulation. IEEE Trans. Inf. Theory **44**(3), 927–946 (1998). https://doi.org/10.1109/18.669123
31. G. Laguna-Sanchez, M. Lopez-Guerrero, On the use of alpha-stable distributions in noise modeling for plc. IEEE Trans. Power Deliv. **30**(4), 1863–1870 (2015). https://doi.org/10.1109/TPWRD.2015.2390134
32. G. De Boni Rovella, GitHub repository. https://github.com/gastondeboni/SVM_for_channel_decoding.git (2024)

Publisher's Note