Channel Equalization Based on Decision Trees

David Felice F. Baptista, Rafael Ferrari, Romis Attux.

***Abstract*—This paper analyzes the application of decision trees to the problem of communication channel equalization. Decision trees are interesting structures because they are nonlinear and relatively simple from a computational standpoint. They are tested for channel models that engender classification tasks of different complexity and have their performance compared to those of the optimal (Bayesian) equalizer and the Wiener linear equalizer. The results are quite encouraging, as they show that the tree-based equalizer reaches, in many cases, a performance similar to that of the Bayesian filter with a computational cost that tends to be lower as the duration of the channel impulse response increases.**

I. Introduction

In simple terms, a communication system is composed of a transmitter, a channel and a receiver. The messages sent by the transmitter are, as a rule, distorted by the channel. Therefore, it is necessary to make use of countermeasures to preserve the relevant information content [1].

One possible strategy is to use a filter – called equalizer – to process the received signal with the aim of "inverting" the effects of the channel [2]. Naturally, this filter must be carefully designed: it should be based on a structure with enough complexity and a criterion for determining the parameters thereof by making use of the available information. From a structural standpoint, equalizers are typically thought of as being, for instance, linear / nonlinear or recurrent / feedforward. As for the adaptation criterion, there are two fundamental possibilities: it can be either supervised or unsupervised [3]. In the former case, there will be labeled training samples to guide the parameter choice, whereas, in the latter, only general statistical properties will be used [4] [5] [6].

The combination of channel, noise and equalization delay may give rise to equalization tasks that demand nonlinear filtering structures. In a supervised scenario, the optimal finite memory equalizer in terms of symbol error rate is the Bayesian equalizer [7]. Although it is theoretically possible to devise design strategies within the Bayesian structural framework, in general, it is simpler and more robust to employ general nonlinear models. Neural networks and Volterra filters are examples of models that were used with success [6].

Decision trees are promising options in nonlinear equalization, but, to the best of our knowledge, there has not been a systematic study concerning their use in this task. The work of Gelfand et al. [8] is conceptually related to this context, but their approach is essentially distinct in that a tree is employed to partition the input space and select a linear equalizer in a piecewise linear scheme, and not to classify the input signal per se. In this work, we will analyze the use of decision trees strictly in the role of equalizers for channel / noise models of different complexity, so as to establish an initial corpus of results and analyses.

In the next sections, we present a brief explanation of the communication channel model, followed by the supervised paradigm of the equalization problem. We also talk about the concept and the induction of decision trees, the test scenario, methodology and obtained results. Finally, we present our conclusions and final remarks.

The authors are with Faculdade de Eng. Elétrica e de Computação - UNICAMP. Emails: davidfelice.ba@gmail.com, rferrari@dca.fee.unicamp.br, attux@dca.fee.unicamp.br. This work was partially supported by CNPq under grant 305621/2015-7. David would like to thank CNPq for partially supporting his work (134058/2016-0). Digital Object Identifier (DOI): 10.14209/jcis.2020.16

II. Communication System Model

A. Information Source

In this paper, the information source will be modeled as a stochastic process generated by a uniform random distribution, in which the symbols are independent and identically distributed (*i.i.d.*) [5].

For the sake of simplicity, the chosen alphabet is $\mathcal{A} = \{-1, +1\}$, which corresponds to a 2-PAM (*Pulse-Amplitude Modulation*) modulation scheme [1]. Nevertheless, the results of this work are extendable to other modulation schemes.

B. Communication Channel Model

Channels are mathematical models of the physical medium through which the modulated information is sent. They are responsible for introducing distortions that affect the information signal. Two of the most usual distortions are *intersymbol interference* (ISI) and *noise* [6].

1) Intersymbol Interference: Intersymbol interference (ISI) is caused by the temporal scattering of the transmitted symbols due to structural limitations of the channel. In this work, we will use a linear ISI mathematical model, which is the most widely employed in practice [4] [6].

ISI can be modeled in terms of a linear combination of delayed versions of the transmitted symbols. In other words, ISI can be understood as the result of a Finite Impulse Response (FIR) filtering process.

2) Noise: Noise is a random-like fluctuation to which the signal is submitted mostly due to thermal agitation and electromagnetic interference [4]. It can be modeled, in many cases, with a signal ρ_k generated by a zero-mean Gaussian stationary stochastic process with variance equal to σ_ρ^2, which is added to the FIR filter output [6].

C. Signal-to-Noise Ratio (SNR)

Signal-to-Noise Ratio (SNR) is a metric that quantifies the relative power of the information signal and of the noise signal

present within it. Mathematically, it is possible to define the SNR as the ratio between the power of the channel output (minus noise) and the noise power (generally in decibels) [9].

III. Communication Channel Equalization

Communication channel equalization is a signal processing technique that has the objective of reversing the distortions introduced by the channel.

A. Channel States

In digital communications, we have the alphabet $\mathcal{A}$ of the transmitted signal as finite and discrete, a fact that implies that $\mathbf{u}$ – derived from the convolution $u[k] = h^*[k] * s[k]$, where $h[k]$ is the impulse response of $H(z)$ – also assumes finite and discrete values, since $H(z)$ is a FIR filter. Given this result, we will further investigate the $\mathcal{U}$ alphabet.

The m dimensional channel states – *i.e.* the states for a $\mathbf{s}$ sequence with size $\eta_c + m - 1$ – consist on the $\mathbf{C} = \{u_0, \ldots, u_{m-1}\}$ set, where $\dim(\mathbf{C}) = l^{(\eta_c+m-1)}$, considering no coincident states. Since each state $\mathbf{c}_i$ is related to a different transmitted sequence – with a total of C states –, we can identify the subset of states which is related to a $s[k-d]$ transmitted symbol.

In this paper, we will work with $m = 2$ and $d = 0$, that is, we will use two samples of $\mathbf{x} = \big[x[k], x[k-1]\big]$ as the equalizer inputs to recover $s[k]$.

B. Supervised Equalization

A supervised equalizer has access to the received symbols $\mathbf{x}$ as well as the transmitted symbols $\mathbf{s}$ [6]. In practice, this is usually done through a predefined training sequence or pilot signal. Figure 1 shows a supervised equalizer block diagram [6]. During the training phase, the equalizer is given the $\mathbf{x}$

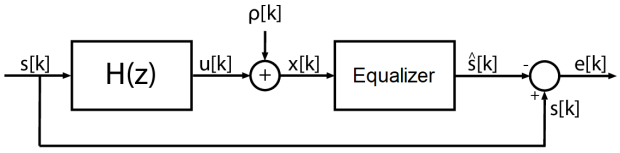

Fig. 1. Supervised equalizer block diagram.

signal and also the $\mathbf{s}$ signal to adapt its parameters so that the error signal $e[k]$, defined by 1, has a magnitude that is as small as possible.

$$e[k] = s[k] - \hat{s}[k] \tag{1}$$

During the operation stage, the equalizer only receives the $\mathbf{x}$ signal and yields $\hat{\mathbf{s}}$, which is the estimate of the $\mathbf{s}$ signal.

From the retrieved symbols, it is possible to calculate the BER for this communication system – channel plus equalizer.

C. MAP Criterion and Bayesian Equalizer

The use of a maximum a posteriori (MAP) criterion to determine the $s[k]$ symbol given the $\mathbf{x}$ vector is optimal with respect to the error probability. Thus, the so-called MAP / Bayesian equalizer is the finite memory symbol-by-symbol equalizer that retrieves the transmitted sequence with the least possible number of classification errors compared to any other equalizer under the same conditions [7].

The Bayesian equalizer can be seen as a classifier that segments the space $\mathbf{x} = \{x[k], \ldots, x[k-m+1]\} \in \mathbb{R}^m$ into $l = dim(\mathcal{A})$ regions, each one corresponding to a $\mathcal{A}$ symbol. For example, for the 2-PAM modulation scheme with $\mathcal{A} = \{-1, +1\}$, and considering a sequence $\mathbf{x} = \{x[k], x[k-1]\}$ with two delays (i.e. $m = 2$), we would have the $\mathbf{x}$ space divided into two regions, one corresponding to $s = -1$ and the other corresponding to $s = +1$. Note that the equalizer output for a given $\mathbf{x}$ would not be exactly $\hat{s} = -1$ or $\hat{s} = +1$, but a possible range of values to be discriminated through a decisor, which sorts this output into $\mathcal{A}$ elements. For this specific example, when we have $\hat{s} = 0$, the decision boundary is obtained.

Considering the 2-PAM alphabet (as will be the rule in this paper) and the defined linear channel model, we identify that any equalizer makes a mistake when the noise is large enough so that a received $\mathbf{x}$ sequence crosses the decision boundary from the correct channel state into any wrong one. Therefore, in order to minimize misclassification, we must then maximize the *a posteriori* probability of the estimated symbol being the correct one given the $\mathbf{x} \in \mathbb{R}^m$ vector received. This assertion can be summarized according to the mathematical expression 2 [7]

$$\hat{s}[k-d] = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, P(s[k-d] = a|\mathbf{x}), \tag{2}$$

where d is a general equalization delay – which in this paper was defined as $d = 0$. The choice of the delay has direct impact on the difficulty of the equalization task for maximum, mixed or minimum phase channels [7].

The optimal equalizer mapping is given by [7]

$$f(\mathbf{x}) = \sum_{j=1}^{2^{\eta_c+m-1}} w_j \exp\left(\frac{-||\mathbf{x} - \mathbf{c}_j||^2}{2\sigma_\rho^2}\right), \tag{3}$$

where $\mathbf{c}_j$ is the j-th channel state for $\mathbf{x} \in \mathbb{R}^m$, $w_j = +1$ for $\mathbf{c}_j$ where $s[k-d] = +1$ and $w_j = -1$ for $\mathbf{c}_j$ where $s[k-d] = -1$ – case of the 2-PAM alphabet.

A few points about the Bayesian equalizer are noteworthy. As one can see, despite the fact that it has the best result among all the equalizers concerning the error rate, for its calculation, it is necessary to have knowledge of the channel states – and, therefore, the h_i coefficients. This is difficult to estimate and often impractical given the eventual time-changing character of the channel.

Another point to consider is the computational complexity. Note that, in (3), the sum goes from 1 to 2^{η_c+m-1}. That is, as the channel size increases, or that of the $\mathbf{x}$ sequence considered for equalization, the number of times that the summation internal expression is calculated increases exponentially – this for each $\hat{s}[k]$ one wants to calculate. Moreover, with this

addition of η_c and/or m, the amount of $\mathbf{C}$ states also increases exponentially (2^{η_c+m-1}), which poses serious computational issues. In fact, the Bayesian equalizer becomes impractical as the length of the channel increases.

D. MMSE Criterion and Wiener Filter

The minimum *Mean Square Error* (MMSE) criterion is based on the minimization of the average square error between the equalizer output and the transmitted signal (aiming at making these signals as similar as possible). It does not take into account the error rate, like the MAP criterion, but, for a linear structure, leads to a straightforward solution [5].

The MSE cost function is given in 4:

$$\begin{aligned} J_{MSE} &= E\big[\ (s[k-d] - \hat{s}[k])(s[k-d] - \hat{s}[k])^*\big] \\ &= E\big[|e(k)|^2\big] \end{aligned} \tag{4}$$

For linear equalizers, the minimization of the cost function 4 leads to a single solution, known as *Wiener* solution [5].

This equalizer – which is termed Wiener Filter – does not require prior knowledge of channel coefficients, as is the case with the Bayesian Equalizer. This means that the technique may lead to a lower computational cost and to a more organic real-time operation.

IV. Decision Trees

Decision Trees are *Machine Learning* methods based on the idea of ramification [10]. The structure is named this way because it is a graph that resembles the image of a tree.

Trees are commonly used for classification, but it is also possible to find in the literature their use in regression tasks [11]. Compared to some AI techniques, they present interesting aspects, like a relatively simple operation and significant interpretability [10] [12] [13]. In the following sections, we will explain in detail how this technique works. In the scope of this paper, we will not use Decision Trees as regression models, so the explanations will be focused on the classification problem.

A. Fundamentals

Trees are supervised *machine learning* methods, which means that their training is based on labeled examples [12].

The operation of Decision Trees assumes that successive divisions of the sample space performed by setting thresholds for the input features are able to determine which class is correct for that example [11]. To determine which attribute will be used for division and its threshold, a classification purity metric is used, which we are going to call *purity gain* (section IV-C) [11].

In the following, one can see a simple example of how a decision tree works, and, in the following sections, we look further into the metrics that can be used to determine, given a tree node, what is the best input feature and what threshold value brings the best sample space splitting.

B. Example

As an example, let us address the following problem: an athletics team needs to select new members. As the number of applicants is too high, they decided to look into the data of their late selection process (table I), and then develop a Decision Tree to make the selection automatically. After the training process, the team arrived at the tree given by the figure 2, which uses as inputs the dominant age, weight, height and dominant hand data of each candidate.

Candidate	Age	Weight	Height	Dom. Hand	Selected?
A	25	65 kg	1.70 m	Right	YES
B	31	63 kg	1.74 m	Left	NO
C	27	51 kg	1.65 m	Left	YES
D	17	75 kg	1.80 m	Right	NO
E	20	60 kg	1.55 m	Left	NO
F	23	49 kg	1.64 m	Right	NO

TABLE I
TABLE FOR THE EXAMPLE IV-B

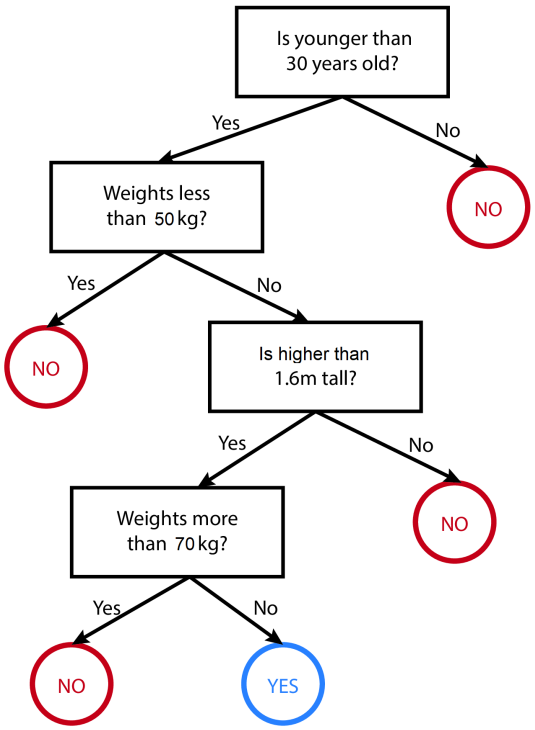

Fig. 2. Decision Tree for the example IV-B.

C. Purity Gain

As seen in the example IV-B, in some cases, it is possible to obtain Decision Trees that perfectly split the samples according to the correct classes. This can be verified by submitting the candidates from the table I to the Decision Tree in figure 2, confirming that all those candidates were correctly classified. For these cases, we say that there is no impurity, hence it is the ideal scenario – except in case of overfitting.

Overfitting is a concept related to an excessive flexibility of machine learning models in comparison with the generative model of the examples presented to them. The more parameters a model has – in the case of a Decision Tree, the more divisions we allow it to perform – the more flexible it is, and can adapt to different situations. Take for example polynomials, which are parametric models. The more parameters are available (degrees of freedom), the larger is the set of functions it is able to approximate [12]. However, a model with many free parameters tends, if proper precautions are not taken, to be subject of overfitting. This corresponds to performing very well for the examples that were used for training, but not for test samples.

Hence, from these assertions, the question arises: how do we choose Decision Trees so that we have the highest possible purity in the final branches? We should then look for purity metrics that allow us to evaluate, before and after the proposed divisions, whether the obtained classification is better or worse than that of the previous situation (whether we have more or less pure sets).

1) Gini Impurity: The Gini Impurity is a metric of class purity defined by the equation 5[14]:

$$Y = \sum_{c \in \mathcal{C}} P[c](1 - P[c]), \tag{5}$$

being S the dataset over which the index is going to be calculated, $\mathcal{C}$ the set of available classes and $P[c]$ is the probability to be the class c given S.

The Gini Gain can be defined as in the equation 6[14]:

$$\begin{aligned} \mathcal{G}_{\mathcal{Y}}(\mathcal{S}) &= Y_p(S) - Y_f(S) \\ &= Y_p(S) - \frac{1}{N} \sum_{b \in \mathcal{B}} n_b Y_b(S_b), \end{aligned} \tag{6}$$

where $Y_p(S)$ is the Gini Index in the branch before the splitting, $Y_f(S)$ is the total Gini Index of the S set (that contains N elements) in the nodes after the branch, $\mathcal{B}$ is the set of leaves originated by the splitting, being $Y_b(S_b)$ the Gini Index for each leaf, taking into account S_b, with n_b elements directed to each leaf[1].

Thus, it makes sense to choose decisions that yield a larger Gini Gain over others that yield a smaller gain. It is the role of the Decision Tree algorithm – presented in section IV-D – to perform impurity gain optimization. As an example, we can compare the Gini Gain for two distinct inputs from the example IV-B, as seen in table II. The Gini Impurity calculated for the root node is given by the equation 7.

[1]To better understand the meaning of these parameters, please refer to the first division of Figure 2 and Table I. $Y_p(S)$ refers to the Gini Index calculated over the set $S = A, B, C, D, E, F$, with $N = 6$; $\mathcal{B}$ corresponds to number of new sets created, in this case, $\mathcal{B} = 2$; $Y_1(S_1)$ corresponds to the Gini Index calculated over the set $b = 1$, originated after the division, i.e., $S_1 = A, C, D, E, F$, with $n_1 = 5$; $Y_2(S_2)$ corresponds to the Gini Index calculated over the set $b = 2$, originated after the division, i.e., $S_2 = B$, with $n_2 = 1$.

Initial Branch	2 YES	4 NO
Gini Index	$Y = 0.444$	

Criterion: **Age**	**>=1.6m**		**<1.6m**	
Total Elements	5		1	
	2 YES	3 NO	0 YES	1 NO
Gini Index	$Y = 0.48$		$Y = 0$	
Gini Gain	$\mathcal{G}_{\mathcal{Y}} = 0.044$			

Criterion: **Dom. Hand**	**Right**		**Left**	
Total Elements	3		3	
	2 YES	1 NO	1 YES	2 NO
Gini Index	$Y = 0.444$		$Y = 0.444$	
Gini Gain	$\mathcal{G}_{\mathcal{Y}} = 0$			

TABLE II
COMPARATIVE GINI GAIN FOR THE PROPOSED 1.6M AGE AND DOMINANT HAND DIVISIONS FROM THE EXAMPLE IV-B

.

$$\begin{aligned} Y &= \frac{2}{6}\left(1 - \frac{2}{6}\right) + \frac{4}{6}\left(1 - \frac{4}{6}\right) \\ &= 0.444 \end{aligned} \tag{7}$$

As one can see, dividing the sample space according to the age input leads to a better Gini Gain, while doing it according to the dominant hand does not help the model to sort the dataset in a purer way.

After this optimization criterion, it is expected that the Decision Tree be able to solve the equalization problem. This assertion is true due to the fact that the equalization problem as stated in section III can be seen as a classification problem, since the desired outcome is to identify which sent symbol, belonging to the alphabet $\mathcal{A}$, a received vector $\mathbf{x}$ is associated with. As so, the BER can be seen as a measure of classification purity, since a misequalized sample can be seen as a misclassified element integrating a set of correctly classified samples, thus, increasing the class impurity. In this sense, the higher the BER, more misclassified elements in the alphabet classes we have, which implies in a higher class impurity.

Since one trains a Decision Tree aiming at the highest possible classification purity – i.e. at the maximization of the purity gain –, it is reasonable to expect that the BER be also, to some extent, minimized. Hence, ideally, it can be expected that the Decision Tree fairly approximate the performance of the Bayesian Equalizer.

D. CART Algorithm

The CART [11] algorithm was developed to induce a Decision Tree, choosing decisions that increase the purity gain within the classification task.

Initially, it is important to define metrics for measuring the performance of the decisions made by the tree. The Gini Gain was taken as the cost function to be maximized, and it was applied greedily through several evaluations performed by the algorithm, which leads us to obtain the attribute – and its respective decision threshold, when applicable – for each division.

The search for a decision threshold – performed when the input variables are continuous – is made through a process of choosing candidates for evaluation. From the dataset received on a particular node, each *feature* value from all examples is sorted. The candidates for decision threshold are then determined as the average between the values of two differently classified examples (see figure 3). After it, the purity gain metric is calculated for all candidates and the one with the highest gain [11][12] is chosen. Finally, the feature – and its threshold – that gave the highest purity gain is used as the node splitter.

Fig. 3. Process of choosing splitting candidates for the example IV-B.

After a first version of the tree generated by the algorithm, a pruning process is started, which evaluates the overall cost function of the tree by removing each of the decisions made – and its subsequent complete branch. If there is an improvement in the cost function or if it remains the same after a pruning, the decision canceling is effective, since it either implied in an improvement or the model shows more parsimony than the previous one – Occam's Razor principle – that is, the model presents a lower computational cost [11] [10] [12].

From the initial node, the Decision Tree divides each node into two new child nodes – based on the division criteria presented above – adding another layer of depth to that branch of the tree. When, for any node, there is no division that presents purity gain or some other criterion is met – such as only one element in the node for new division – that node is no longer divided and is considered as a leaf [11] [10].

A pseudocode of the CART algorithm – concerning the equalization problem presented in this paper – is presented in the algorithm 1, where $\mathcal{U}$ is the whole training data set.

Algorithm 1 CART Pseudocode

Initializing:
Start $\mathcal{L}$ as a FIFO stack
$\mathcal{L} \leftarrow \mathcal{U}$
Splitting:
while $\mathcal{L}$ is not empty **do**
    pop $\mathcal{B}$ from $\mathcal{L}$
    for each feature F_i from the inputs in $\mathcal{B}$ **do**
        sort F_i
        find the threshold candidates T
        for each T_i **do**
            calculate the information gain after T_i division
    Divide $\mathcal{B}$ into $\mathcal{B}_1$ and $\mathcal{B}_2$ according to the pair $\{F_i, T_i\}$ that provides the greater information gain
    for each $\mathcal{B}_i$ **do**
        calculate the purity metric for $\mathcal{B}_i$
        if stopping criteria is not met **then**
            push $\mathcal{B}_i$ into $\mathcal{L}$
prunning:
calculate the purity metric P_f for $\mathcal{U}$ subimmited to the induced tree
for each division D_i **do**
    remove D_i
    calculate the purity metric P_g for $\mathcal{U}$ submitted to the pruned tree
    if $P_g \geq P_f$ **then**
        make the removal D_i definitive

$x[k]$	$x[k-1]$	$s[k]$
-1.501	-1.497	-1
-1.499	-0.502	-1
-0.500	0.500	-1
-0.502	1.498	-1
0.503	-1.501	1
0.500	-0.500	1
1.504	0.498	1
1.500	1.502	1

TABLE III
INPUTS DATA SET TO THE DECISION TREE TRAINING.

E. Example: Application to the Channel Equalization Problem

In order to show the applicability of the Decision Tree to the problem of channel equalization, a Decision Tree model will be trained according to the presented CART algorithm (section IV-D), to perform the equalization of a simple channel.

The process of obtaining the trained model aims at developing the reader intuition on the modus operandi in more complex situations. Therefore, the channel $h = [1\ 0.5]$ was chosen, with a tiny amount of noise applied – just for the sake of illustration, since in the test scenarios there was noise applied according to the SNR scenario presented –, null delay and the number of equalizer inputs equal to two ($m = 2$), which generates the data set presented in the table III, to be passed on to the model as inputs. For didactic purposes, the presented examples are representative of all channel states, and, therefore, also of the entire problem universe. In practice, datasets often have more than one example for each state. However, there may be some scenario in which the number of examples is insufficient to represent all states, compromising the quality of the model.

The CART stopping criteria defined here are pure leaves (Gini Impurity equals $Y = 0$) or minimum number of samples in node of one element.

Initially, it is important to calculate the Gini Impurity of the root node (initial branch). After this value, we can calculate the Gini Gain of each possible division made by the Tree, allowing us to evaluate what are the optimal attributes and their respective thresholds for each division. In equation 8, we have the Gini impurity for the root node.

Split	Left Branch		Right Branch		Gini Gain
	$\hat{s}[k]=-1$	$\hat{s}[k]=1$	$\hat{s}[k]=-1$	$\hat{s}[k]=1$	
$x[k]=0$	4	0	0	4	$G_Y=0.5$
	$Y_b=0$		$Y_b=0$		
$x[k-1]=-1.499$	1	0	3	4	$G_Y=0.255$
	$Y_b=0$		$Y_b=0.490$		
$x[k-1]=-0.501$	2	1	2	3	$G_Y=0.038$
	$Y_b=0.445$		$Y_b=0.480$		
$x[k-1]=0.499$	3	2	1	2	$G_Y=0.038$
	$Y_b=0.480$		$Y_b=0.445$		
$x[k-1]=1.500$	4	3	0	1	$G_Y=0.255$
	$Y_b=0.490$		$Y_b=0$		

TABLE IV
GINI GAIN CALCULATIONS TO THE DIVISION CANDIDATES.

.

$$Y_p = \frac{4}{8}\left(1-\frac{4}{8}\right)+\frac{4}{8}\left(1-\frac{4}{8}\right) \\ = 0.5 \tag{8}$$

The next step of the algorithm is to define what will be the feature of the model first division and the corresponding threshold. To perform this task, according to the CART algorithm, one must sort the values of the input attributes and choose as candidates for division the mean value between differently classified adjacent samples (see figure 4). Knowing the candidates, one should calculate the Gini Gain for each of them and choose as the best division the one that brings the highest purity gain. These calculations were performed and are presented in table IV.

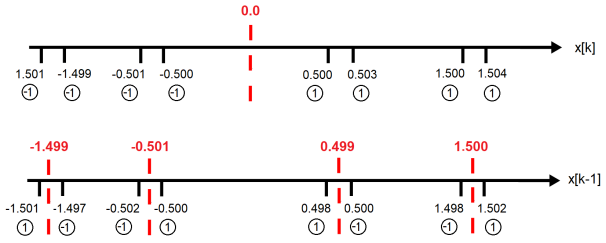

Fig. 4. Division candidates and their thresholds (red).

As follows from table IV, the division with the highest Gini Gain is $x[k]=0$, leading to two pure leaves, which meets one of the established stopping criteria.

Thus, we have, as a final version, the tree in figure 5.

In order to demonstrate the reliability of the example to the implemented CART algorithm, a Decision Tree was trained for the same channel, with 100,000 randomly generated samples, without noise, obtaining the model shown in figure 6.

As one can notice, the tree model obtained by the CART algorithm is equivalent to the one discussed in the example IV-E.

V. METHODOLOGY

In this paper, we will use the Decision Tree – as described in section IV – as an equalization method expected to perform better than the Wiener Filter and, hopefully, analogously to the Bayesian Equalizer.

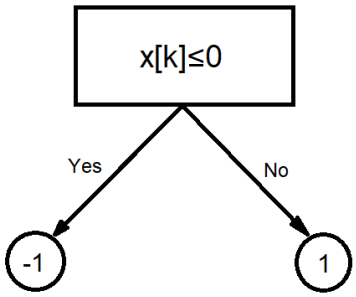

Fig. 5. Final version of the trained Decision Tree to the example IV-E.

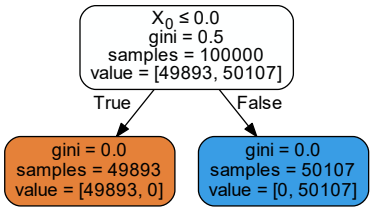

Fig. 6. Decision Tree presented by the implemented CART algorithm, where x_0 represents $x[k]$.

As already stated, the alphabet chosen is 2-PAM and all techniques will constitute supervised equalizers with null delay ($d=0$) and $m=2$ – i.e., the inputs are $\mathbf{x}=[x[k],x[k-1]]$ aiming to recover $s[k]$ – so it is possible to compare them directly between themselves.

For the Decision Tree and Wiener filter, 85,000 examples were presented for training each SNR scenario, and 15,000 examples for test. To avoid the effect of suboptimal solutions, 10 trials of these models were considered; the Bayesian Equalizer is deterministic and does not require any training; the methods were evaluated until at least 100 errors were committed. The BER was calculated over all test examples presented to the methods.

The SNR values will be defined for each test scenario individually, since the state topology has high influence in the noise immunity for each case.

VI. TEST SCENARIO

In this section, we discuss the characteristics of the channels we presented to the techniques proposed. It was chosen four representative scenarios, such as minimum-phase, maximum-phase and mixed-phase channels, also varying the length (η_c) of them.

Table V shows the main attributes of each scenario and the configuration of their channel states is presented in figure 7.

A. Channel 1 - Minimum-phase

Channel 1 represents a minimum-phase short scenario, which implies that all its zeros are inside the Unit Radius

Channel	Coefficients	η_c	No. of States
1	$\mathbf{h} = [0.894\ 0.447]^T$	2	8
2	$\mathbf{h} = [0.447\ 0.894]^T$	2	8
3	$\mathbf{h} = [0.5\ 0.71\ 0.5]^T$	3	16
4	See eq. 9	10	2048

TABLE V
CHARACTERISTICS OF THE PROPOSED TEST SCENARIOS.

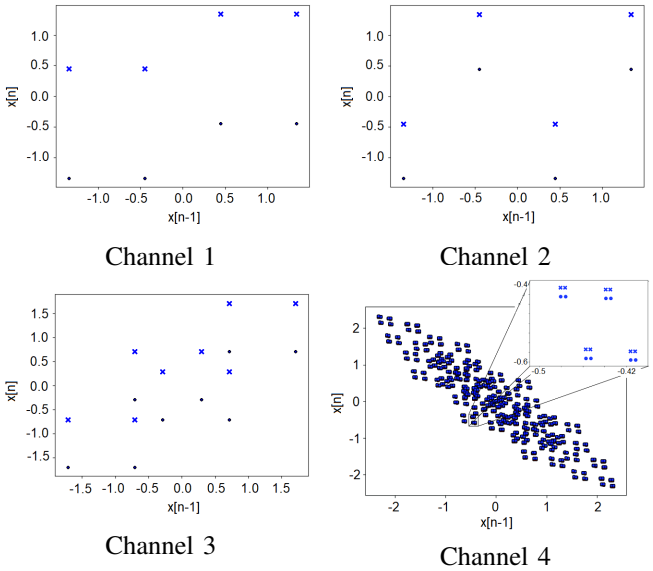

Fig. 7. States of the channels proposed as scenarios. The states represented with '×' corresponds to $s[k] = +1$ and the ones represented with '○' corresponds to $s[k] = -1$. The Channel 4 figure is replicated in figure 8 for better visualization.

Circle. For the equalization topology and delay chosen, this channel has linearly separable states (see figure 7-(Channel 1)), which is expected to present satisfactory results for all equalizers, including the Wiener Filter, as a linear technique. For this case, it is going to be tested a SNR scenario as follows: SNR $= [13\text{dB}, 10\text{dB}, 7d\text{dB}, 4\text{dB}, 1\text{dB}]$.

B. Channel 2 - Maximum-phase

Channel 2 represents a maximum-phase short scenario, with all its zeros outside the Unit Radius Circle. This channel has the same amount of states of Channel 1, but not linearly separable (check figure 7-(Channel 2)), implying that it is impossible for the Wiener Filter to achieve BER=0, even in a noiseless situation – it requires a nonlinear equalization boundary. For Channel 2, the SNR scenario will be SNR $= [16\text{dB}, 13\text{dB}, 10\text{dB}, 7\text{dB}, 4\text{dB}, 1\text{dB}]$.

C. Channel 3 - Mixed-phase shorter-term

Channel 3 is a mixed-phase scenario, with 16 states. The equalization boundary required to perfectly separate is non-linear (see figure 7-(Channel 3)), and its zeros are located on the Unit Radius Circle, which makes the equalization task considerably more difficult than the past two scenarios. Channel 3 will be tested for a SNR scenario as follows: SNR $= [16\text{dB}, 13\text{dB}, 10\text{dB}, 7\text{dB}, 4\text{dB}, 1\text{dB}]$.

D. Channel 4 - Mixed-Phase longer-longer

We define Channel 4 as a longer-term channel, since its length η_c is large enough so that the computational cost of the Bayesian equalizer may be prohibitively high for real time applications. It is a mixed-phase channel, with 2048 states and a highly nonlinear boundary is required for its equalization. In equation 9, it is presented its coefficients and figure 8 shows the configuration of its channel states.

$$\begin{aligned}\mathbf{h} = \big[&0.0114693\ \ -0.0802852\ 0.2565113\ \ -0.4875893\\ &0.6008345\ \ -0.4947074\ 0.2725110\ \ -0.0972204\\ &0.0206233\ \ -0.0021290\big]^T.\end{aligned} \tag{9}$$

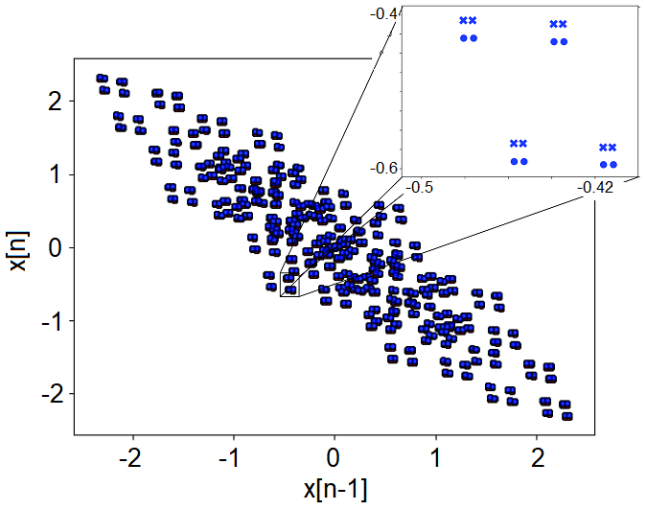

Fig. 8. This figure represents the Channel 4 states. The states represented with '×' corresponds to $s[k] = +1$ and the ones represented with '○' corresponds to $s[k] = -1$.

As can be seen in figure 8, for this scenario, there are several adjacent clusters of states with different classifications, implying that the equalizer decision boundary must be flexible enough to make the necessary folds.

A larger m may improve the performance of the equalizer when the channel response is longer, as more information may be required for the model to perform well. Despite of that, as mentioned in the section V, $m = 2$ is going to be constant for all test scenarios, for the sake of comparability between the analyzed methods.

This scenario is of special interest since its features, the high amount of states and their topology, are expected to be challenging for the Wiener filter and will lead to a high computational cost for the Bayesian Equalizer. It is expected that the Decision Tree may stand out as a technique with satisfactory performance and acceptable computational cost, especially in *online* applications. The SNR scenario presented will be SNR $= [70\text{dB}, 60\text{dB}, 50\text{dB}, 40\text{dB}, 30\text{dB}]$.

VII. RESULTS

A. Performance of the Methods

1) Channel 1: In the figure 9 and table VI are presented the results of the proposed equalizers to Channel 1.

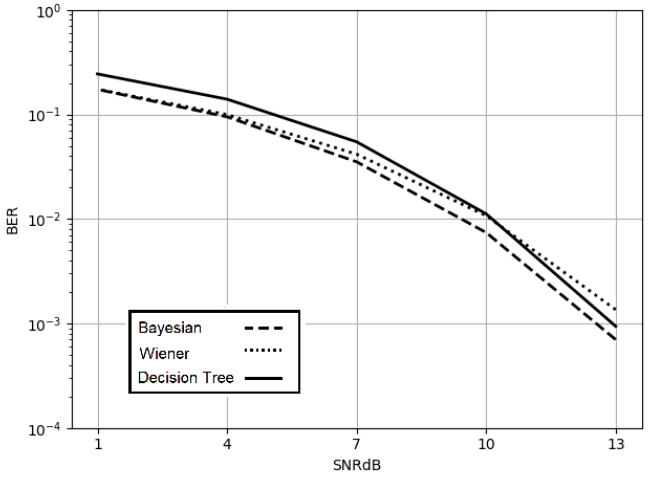

Fig. 9. Performance of the methods for the Channel 1 (BERxSNR).

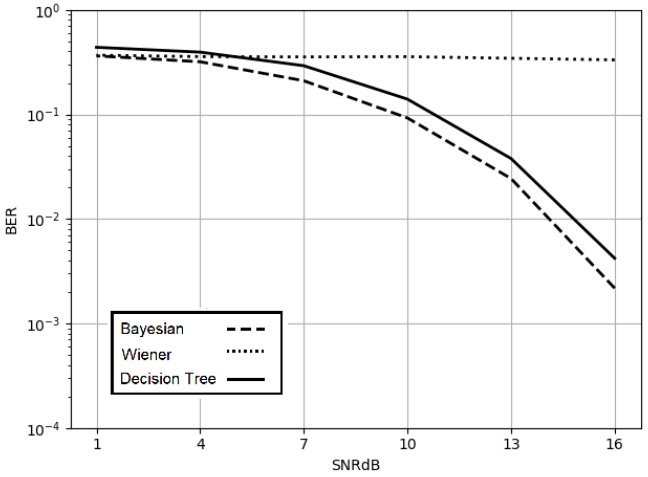

Fig. 10. Performance of the methods for the Channel 2 (BERxSNR).

SNR (dB)	Bayesian	Wiener	Decision Tree
13	0.0007	0.001	0.0009
10	0.007	0.011	0.011
7	0.035	0.042	0.055
4	0.095	0.100	0.140
1	0.174	0.174	0.244

TABLE VI
MEAN ERROR RATE FOR CHANNEL 1, EXPRESSED IN ERRORS BY SENT SYMBOL.

As expected, the Wiener Filter presents good results. The method approximated the Bayesian Equalizer, that, as previously mentioned, is the optimal finite memory method concerning the bit error rate (BER). The Decision Trees also present applicability, since approximated the Bayesian Results as well, especially concerning the high SNR scenarios.

For the noisier scenarios (7dB, 4dB and 1dB), the Wiener Filter outperformed the Decision Trees results, fact that can be explained due to the *overfitting* that affect the Decision Trees, as stated in the section IV-C. In a noisy scenario, the proposed Decision Tree tends to select a high amount of outliers at the expense of learning a general rule for separating the state clusters, loosing performance.

2) Channel 2: For Channel 2, the performance of the analysed equalizers can be seen in the figure 10 and table VII.

SNR (dB)	Bayesian	Wiener	Decision Tree
16	0.002	0.332	0.004
13	0.024	0.344	0.038
10	0.092	0.357	0.140
7	0.210	0.355	0.292
4	0.319	0.357	0.394
1	0.363	0.369	0.438

TABLE VII
MEAN ERROR RATE FOR CHANNEL 2, EXPRESSED IN ERRORS BY SENT SYMBOL.

Since this scenario requires a nonlinear equalization boundary due to the states topology, as showed in figure 7-(channel 2), the linear Wiener filter presents poor performance, even for the least noisy scenario (16 dB).

On the other hand, the Decision Tree approximated the Bayesian Equalizer results. It is possible to notice the same tendency for both methods, and the characteristic loose of performance in the noisy scenarios due to the overfitting. One more time, the Wiener Filter outperformed the Decision Tree in the lowests SNR scenarios.

3) Channel 3: In figure 11 and table VIII are presented the results of the equalizers for the Channel 3.

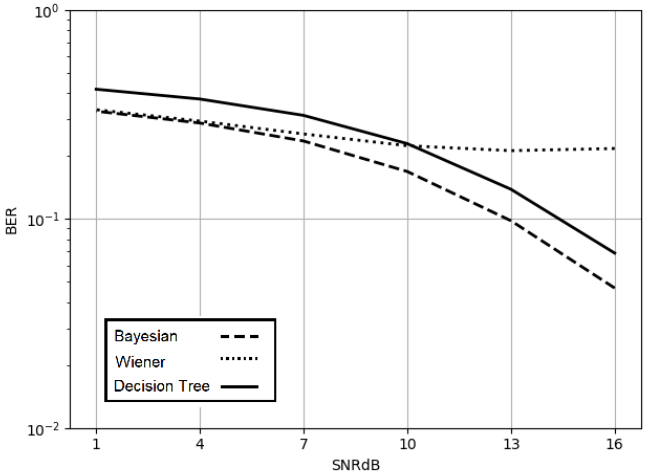

Fig. 11. Performance of the methods for the channel 3 (BERxSNR).

SNR (dB)	Bayesian	Wiener	Decision Tree
16	0.048	0.217	0.069
13	0.098	0.212	0.139
10	0.169	0.224	0.229
7	0.237	0.255	0.313
4	0.288	0.294	0.375
1	0.328	0.333	0.417

TABLE VIII
MEAN ERROR RATE FOR CHANNEL 3, EXPRESSED IN ERRORS BY SENT SYMBOL.

As stated in the VI-C, Channel 3 is a difficult scenario to perfectly equalize, since its zeros are on the Unit Radius Circle, which implies in a intricate state topology (see figure 7-(channel 3)), requiring a nonlinear separation boundary for the chosen equalization delay. In this sense, as happens for Channel 2, the Wiener Filter linear equalization boundary is not well suitable for the present scenario. Its results were around BER=0.22 and BER=0.33, being outperformed in almost 315% more errors than the Decision Tree for the least noisy scenario (16dB).

Due to its lower noise immunity, the BER attained by the nonlinear methods for Channel 3 is considerably higher comparing to Channels 1 and 2. The Decision Tree presents poorer results than the Wiener Filter in the three lowests SNR values. The low noise immunity implies in a higher amount of outliers, increasing the effect of overfitting for the Decision Trees. Despite of this fact, it once more approximated the Bayesian Equalizer results for the higher SNR scenarios.

4) Channel 4: The channel presented here is of special interest, since it has ten coefficients (longer-term) and is of mixed phase. Finding techniques that perform well on difficult scenarios and still can be computed with relatively low computational cost is an interesting subject for real-time systems. In the figure 12 and table IX, we present the results for this scenario.

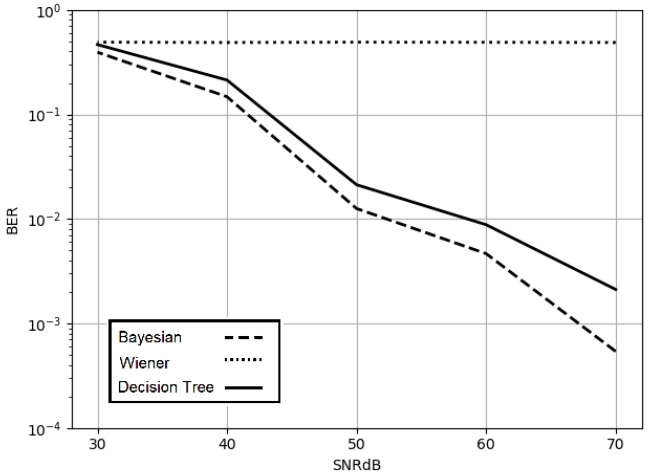

Fig. 12. Performance of the methods (BERxSNR).

SNR (dB)	Bayesian	Wiener	Decision Tree
70	0.0005	0.488	0.002
60	0.005	0.490	0.009
50	0.013	0.491	0.021
40	0.148	0.487	0.213
30	0.393	0.491	0.465

TABLE IX
MEAN ERROR RATE, EXPRESSED IN ERRORS BY SENT SYMBOL.

Analyzing the Wiener Equalizer performance, its unsuitability to this scenario is patent. Regardless of the low SNR, the method presented BER close to 0.5, the worst possible result. Once again, the performance can be explained by the fact that the linear equalizer can produce only linear decision boundaries while the problem requires a nonlinear boundary to classify the signal appropriately.

In the analysis of the proposed method – Decision Tree –, the results are quite reasonable. The Decision Tree approximated the optimal equalizer trend. As the noise increases, the gap between the performance of the Decision Tree and the Bayesian equalizer increases, but the Decision Tree has better results than those of the Wiener Filter for the the entire SNR range considered in the experiment.

B. Decision Tree Computational Complexity

From the previous results indicating the good performance of the Decision Tree, it is important to evaluate the difference in computational cost between the tested algorithms, aiming to identify if computational advantages arise by employing one over the other. If it is the case, this information will allow us to understand the applicability of the Tree in *online* contexts.

Since the Decision Tree is a nonparametric method, whose decision structure is built after an optimization process that only uses examples from the presented data set (and some structural assumptions), for each set of samples – even considering the same channel and parameters –, a different tree can be developed, implying in a distinct computational cost for its execution. However, it is possible to define some hypotheses of correlation between the tree execution complexity and some aspects of the channel equalization problem.

Since it is likely that some parameters of the channel equalization problem – as channel length and state distribution – affect the computational complexity of the Decision Tree, it is necessary that different communication channels, with different characteristics, be evaluated. Hence, for these tests, the scenarios presented in section VII-A will be evaluated.

In this work, the decision tree maximum depth is going to be used as the evaluation cost criterion. The maximum depth consists on how many comparisons are needed to classify an input vector that travels the longest path within the model structure. Hence, it is possible to guarantee that any other samples presented to the Tree are calculated with less comparisons, implying a faster execution.

It is due to the decision tree model structure that the more fragmented the Tree decision boundary is – i.e. the more segmented the input space is – the more comparisons are necessary for its execution. This statement follows from the fact that a greater boundary segmentation implies on

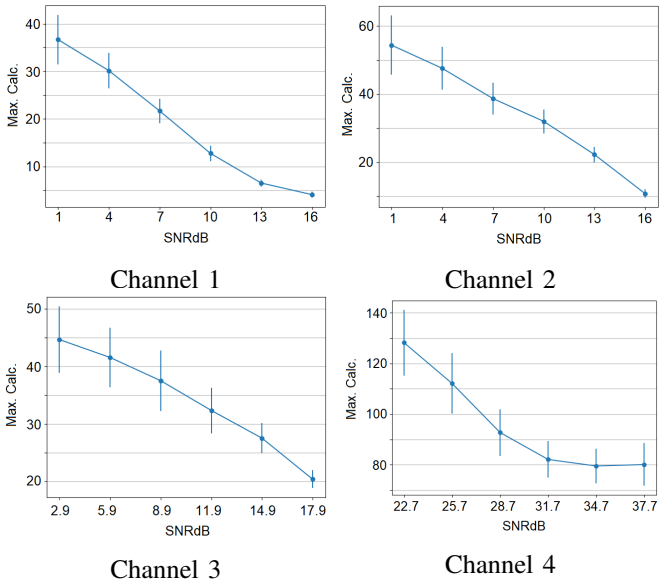

Fig. 13. Mean of the maximum number of comparisons for the Decision Tree evaluation.

more non-contiguous regions of classification, requiring more comparisons to discriminate which region an input vector belongs to.

It is expected that a higher number of channel states, associated with its zeros distribution and an equalization delay that implies a complex decision boundary, cause a higher computational cost for the tree.

Also, due to the segmentation of the Tree decision boundary, it is possible that there may be a relation of its higher computational cost with the noise power increase. As follows from model performance results, the Decision Tree tends to overfit in noisy scenarios. Its flexibility and non-parametric characteristics are essential to this phenomenon, allowing the model to select *outliers* and segment the input space in a way that those specific samples are classified according to their correct class.

As one can notice, the Decision Tree evaluation cost has the potential to be quite casuistic and dependent on factors other than just the channel length – as it is determinant for the Bayesian Equalizer – or the power of noise. Thus, some computational simulations are presented to assess the hypothesis described above. In the experiments, the mean and standard deviation of the maximum number of comparisons were calculated concerning a hundred different Tree evaluations, each of them trained with at least 250 samples per channel state.

Regarding the amount of comparisons vs. the order of the channel, it is observed in Figure 13 that the results confirm the hypothesis. Observing the less noisy scenario, it can be noted that channel 4 had the highest number of comparisons, followed by channel 3, channel 2 and, finally, channel 1. Also as presented, despite the fact that both channels 1 and 2 had the same number of states, the first one is linearly separable, favoring a smaller number of comparisons in relation to channel 2, which has a more complex decision boundary and a higher maximum number of comparisons for its evaluation.

Analyzing the noise influence on the computational cost, it can be noticed that, as expected, the maximum tree depth becomes larger as the noise power increases. However, as also suggested, depending on the noise power and the arrangement of channel states, there may be a decrease in the Decision Tree cost with the noise increase. This is observed when calculating the evaluation cost for channel 4, with noise scenarios such as SNR = 70dB (less noisy), resulting in 167.9 ± 13.2 comparisons and SNR = 30dB (more noisy), resulting in 90.7 ± 8.4 comparisons. From the experiments, it is not possible to categorically state the cause for this phenomenon, assuming that it is due to a better adjustment of the optimization algorithm, resulting in less segmented decision regions given the higher noise power. For a safer statement concerning the reasons for this event, further study concerning the Decision Tree optimization algorithms in different channel and noise scenarios are required.

The Bayesian Equalizer computational cost is proportional to the number of channel states which, in turn, increases exponentially with the channel length and with the number of filter inputs, as discussed in section III-C. Therefore, the computational cost of the Bayesian Equalizer is related to the evaluation of 8 exponentials in Channels 1 and 2, 16 exponentials in Channel 3 and 2048 exponentials in Channel 4. In some cases, mainly for lower SNR values in shorter channels, the maximum number of comparisons of the tree is larger than the number of channel states. However, it is important to note that, especially for channel 4, the Decision Tree evaluation cost is considerably lower than the Bayesian Equalizer. In the worst case, the maximum number of comparisons for the tree was around 128.3 ± 1.0 calculations. When considering the Bayesian Equalizer equation (eq. 3), for the same scenario, its computational cost is dominated by the calculation of 2048 exponentials, which computation is individually already more expensive than a comparison.

Also, from the evolution of the most costly scenarios for each channel, it is possible to suggest that the maximum number of tree comparisons does not appear to be exponentially related to the order of the equalized communication channel – as it is the case for the Bayesian – since from channel 3 to channel 4 we increased 128 times the number of states, and only about three times more comparisons were required to evaluate the largest branch of the Tree. This phenomenon can be due to the Decision Tree topology: at each branch, it subdivides the input space into two, and so it is possible, concerning a linear depth, to analyze an exponential space of possibilities.

Concerning the Wiener Filter, it can be stated that this technique is computationally cheaper than both the Decision Tree and Bayesian Equalizer. Its evaluation is dominated by as many multiplications as the filter order. For the cases here presented, its evaluation is made after two multiplications and one sum, being the least expensive of all.

VIII. Conclusion

The present paper results from the perspective of using *Decision Trees* to solve the problem of *Communication Channel*

Equalization. As proposed, it was thought that this technique could be an alternative to the optimal equalizer with lower computational cost, to be used in *online* systems.

To comparatively analyze the tree performance, the Wiener Filter was employed as a benchmark technique, with low computational cost and usable in online systems. A great advantage of it is the lack of prior knowledge of the channel coefficients, parameters that are difficult to estimate, specially for time-variant channels. For the presented channels and equalization delay, its performance was poor for the majority of the scenarios, except for Channel 1, which is suitable for its approach.

Another proposed comparison method was the Bayesian Equalizer. Being the optimal finite memory method concerning the bit error rate, it actually presented the best results for all tested SNRs scenarios. Acting as the theoretical lower limit for other techniques, the Bayesian Equalizer has two major disadvantages that prevent it from being used in practical scenarios: it has high computational complexity – non-polynomial – and also requires estimation of channel coefficients. As the channel has a longer temporal response, the number of calculations to be done for each output grows exponentially – see equation 3 – quickly becoming unfeasible.

The Decision Tree performance for the equalization problem can be considered quite satisfactory. Given the simplicity of training, low computational complexity and no prior knowledge of channel coefficients, its usage is quite appropriate for online systems. When it comes to performance, especially considering the higher SNR scenarios, its performance was encouraging. With results close to the Bayesian Equalizer – and far better than the Wiener filter –, the Decision Tree can be considered as a tool with potential for effective use. As for the scenarios with lower SNRs, the technique tends to overfitting and lose performance.

Especially considering the longer-term channel as test scenario, the Decision Tree exhibited excellent results. Channels like this are difficult to solve, especially in a real-time paradigm. As noted by the channel states, the complexity of the decision boundary topology that reasonably separates the states is highly nonlinear, leading the Wiener method to yield results very close to the worst possible. The fact that the Decision Tree performs considerably better than this technique – approaching the Bayesian Equalizer result – with low computational complexity shows an important advance in the search for long channel equalization methods.

Yet, it is important to notice that the computational complexity of the Decision Tree (as a channel equalizer) is somehow casuistic, but, apparently, there is no exponential relationship between the number of channel states, or equalizer inputs, and the number of comparisons required to evaluate the Tree – as it does for the Bayesian Equalizer. In order to confirm such assertion, further studies are needed. Nevertheless, the Monte Carlo simulations performed in the session VII-B indicate that the tree may approximate the optimal performance with reduced computational cost.

ACKNOWLEDGMENT

The authors thanks the Prof. Dr. Roberto A. Lotufo for the support and insight in the usage of Decision Trees in the Long-Term Equalization problem. This work was partially supported by CNPq (305621/2015-7 and 134058/2016-0).

REFERENCES

[1] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital Communication: 3rd Edition*. Springer Science & Business Media, 2003.
[2] R. W. Heath Jr., *Introduction to Wireless Digital Communications*, 1st ed. Prentice Hall, 2017.
[3] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Springer, 2012.
[4] S. Haykin and M. Moher, *Communication Systems*, 5th ed. Wiley, 2009.
[5] S. Haykin, *Adaptive Filter Theory: 4th Edition*. Prentice Hall, 2001.
[6] J. M. T. Romano, R. Romis Attux, C. C. Cavalcante, and R. Suyama, *Unsupervised Signal Processing: Channel Equalization and Source Separation*, 1st ed. CRC Press, 2016.
[7] S. Chen, B. Mulgrew, and S. McLaughlin, "Adaptive bayesian equalizer with decision feedback," *IEEE Transactions on Signal Processing*, vol. 41, no. 9, pp. 2918–2927, 1993.
[8] S. B. Gelfand, C. Ravishankar, and E. J. Delp, "Tree-structured piecewise linear adaptive equalization," *IEEE transactions on communications*, vol. 41, no. 1, pp. 70–82, 1993.
[9] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, 1st ed. Waveland Pr Inc, 1990.
[10] R. O. Duda, D. G. Stork, and P. E. Hart, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2000.
[11] L. Breiman, *Classification and regression trees*. Routledge, 2017.
[12] C. M. Bishop, *Pattern recognition and Machine Learning*. Springer, 2006.
[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn v0.21.3: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

David Felice F. Baptista PhD candidate in Energy Regulatory Affairs in University of Campinas (UNICAMP). Electrical Engineer (2015) and Master in Electrical Engineering focused in Computer Engineering (2019) from University of Campinas (UNICAMP). Law School student in Campinas Catholic University.

Rafael Ferrari Rafael Ferrari received his B.S. (2001), M.Sc.(2005), and Ph.D. (2011) degrees in Electrical Engineering from the University of Campinas (UNICAMP). From 2011 to 2015, he has been a researcher at the Center for Petroleum Studies (CEPETRO-UNICAMP). Currently, he is an Assistant Professor at the School of Electrical and Computer Engineering (FEEC) of UNICAMP. His main research interests include computational intelligence and digital signal processing, especially their application to communication systems and to seismic data analysis.

Romis R. de F. Attux Romis Attux was born in Goiânia in 1978. He received the titles of Electrical Engineer (1999), Master in Electrical Engineering (2001) and Doctor in Electrical Engineering (2005) from the University of Campinas (UNICAMP). He is currently an associate professor at the same institution. His research interests are: adaptive signal processing, machine learning, brain-computer interfaces, chaotic systems and ethics in artificial intelligence.