ADAPTIVE CHANNEL EQUALIZATION USING CLASSIFICATION TREES

Taneli Haverinen, Arto Kantsila, Mikko Lehtokangas, Jukka Saarinen
Digital and Computer Systems Laboratory, Tampere University of Technology,
P.O.Box 553, FIN-33101 Tampere, FINLAND
e-mail: tanelih@cs.tut.fi

ABSTRACT

This paper focuses on adaptive equalization of binary signals in a baseband digital telecommunication system. Equalization and detection are considered as a classification problem. Fixed-length sequences of received observations form a multidimensional signal space, which can be partitioned using the proposed classification tree algorithm. Top-down approach is used in tree induction, and splitting is done based on information gain criterion. Overfitting is avoided by utilizing a pruning algorithm. The advantages of this method are its simplicity and straightforwardness and thereby the reduction of computational complexity compared to other well performing equalizers. Experimental results and comparison with a cascade-correlation trained multilayer perceptron neural network equalizer are given.

INTRODUCTION

Radio channels are typically time-variant and the channel impulse response is unknown. To mitigate intersymbol interference (ISI) resulting from multipath propagation of the signal, a variety of adaptive equalizers have been developed and introduced. The maximum likelihood sequence estimation implemented by the Viterbi algorithm [1] has been considered as an optimum method to combat the ISI, but it also involves high computational complexity. On the other hand, linear equalizer is easily implemented using an adaptive least mean square (LMS) algorithm [2], but its performance is unsatisfactory in the case of nonlinear ISI or deep spectral nulls in the channel frequency response. The efficiency of a linear equalizer can be improved to some extent by removing part of the ISI from the present estimate as is done in decision feedback equalization [3].

The methods given above represent the traditional way of viewing equalization from a communication theoretic point of view. Our approach is to consider equalization and detection as a classification problem. Given the observation sample sequences of length *K*, each sequence maps to a point in the *K*-dimensional signal space. Detection is accomplished by partitioning the signal space into appropriate decision regions. Neural networks are known to be superior classifiers, but they involve a considerable amount of learning by adjusting the weights. Classification trees are able to solve linearly non-separable problems by placing consecutive axis-parallel decision boundaries in a straightforward manner. The proposed classification tree method significantly reduces the computational load while still offering performance comparable to neural networks.

SYSTEM MODEL

Our work is loosely based on the Global System for Mobile Communications (GSM) [4]. The information signal consists of bursts of binary symbols taking values of either 1 or –1. Each burst begins with a training sequence, which is 26 bits in length and known to the receiver. Therefore it can be used to adapt the equalizer. The latter part of the burst contains 116 bits of data payload, which is not known to the receiver. The total length of the burst is 142 bits, and the burst is oversampled at the transmitter by a factor of 3. The information signal is transmitted as two-leveled baseband signal.

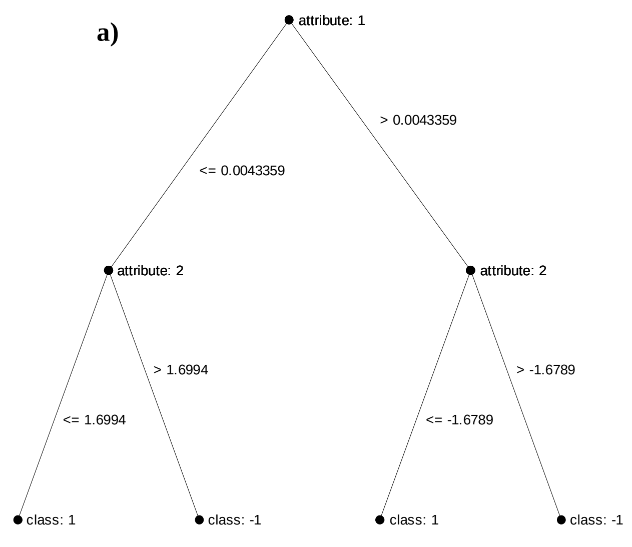
The communication channel used in simulations has both ISI and additive white Gaussian noise. The ISI part, resulting from multipath propagation, is modeled as a finite impulse response (FIR) discrete-time filter. We can formalize the relationship between the channel outputs y and the transmitted binary signal a with the following equation

$$y_n = \sum_{i=0}^{N} h_i a_{n-i} + \delta_n \tag{1}$$

where h_i are the channel coefficients and δ represents additive noise.

CLASSIFICATION TREE EQUALIZER

Our classification problem is to generate a predictive function which takes a selected portion of channel outputs y as attributes and returns an estimate of the class of the object, i.e. the transmitted symbol a_n, as its output. Classification trees perform the prediction by applying a sequence of tests on the attributes [5]. In our application these tests are univariate, i.e. only one attribute at a time is tested, and a sequence of tests can be represented as a binary tree. The outcome of the test determines which



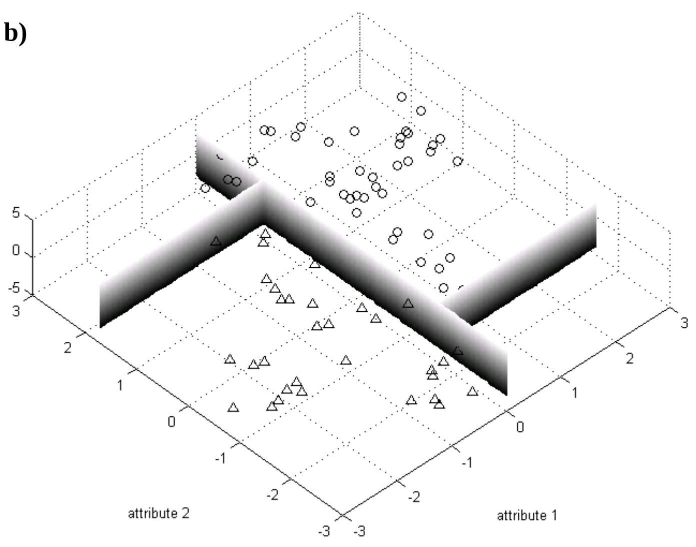


Figure 1: a) Simple binary tree with three splitting nodes and four leaves, which are attached with a class label. **b**) Three-dimensional signal space representation of that same tree.

branch to take when proceeding to the next level of the tree. Univariate tests produce axis-parallel decision boundaries and progressive partitioning of the signal space as consecutive tests are performed from the root to a leaf. Each leaf is attached with a prediction $\hat{a}$ of a class label a. An illustrative example is shown in Fig. 1.

Before making use of a tree it must be generated using a set of training examples and a suitable tree induction algorithm. We have adopted so-called top-down approach [5]. Growing of the tree begins with one single node, the root, to which the whole set of training examples is assigned. These examples are divided into two subsets by creating and applying a test on the attribute values, according to some splitting criterion. Let us give a notation *split* for an {attribute, treshold level} -combination. From all reasonable *splits* the algorithm picks the one, which by some greedy criterion most efficiently separates objects belonging to different classes. Then the algorithm advances to the next level and performs exactly the same procedure on both of the example subsets. Each time a test is attached to the corresponding node. This splitting of the example data is continued until all the training examples at a node belong to the same class or have exactly the same attribute values.

We have used information gain criterion in the tree-growing phase [6]. Information gain is a measure of the expected reduction in entropy caused by partitioning the training set according a particular *split*. In case of target classes -1 and 1, entropy of the collection of training examples S is defined as

$$Entr(S) = -p_{-1}\log_2 p_{-1} - p_1 \log_2 p_1 \qquad (2)$$

where p_{-1} and p_1 are the proportions of the two classes in S. Low entropy implies good homogeneity within the examples belonging to a same cluster, and therefore the *split* resulting with the best information gain is used as a splitting test. The information gain of *split*, relative to S, can be expressed for binary trees as

$$Gain(S, split) = Entr(S) - \frac{|S_l|}{|S|} Entr(S_l) - \frac{|S_r|}{|S|} Entr(S_r) \qquad (3)$$

where S_l and S_r are the left and the right subset of S according to *split*. For discrete-valued attributes finding the set of candidate *splits* would be unambiguous, but our attributes are all continuous-valued. A common practice is to place candidate thresholds dynamically midway between adjacent examples that differ in their target classification.

To avoid the problem of overfitting, we have implemented a pruning algorithm. Pruning means converting some internal nodes into leaves by collapsing the subtree rooting at them and assigning class labels to them [5]. We have used MDL-SLIQ algorithm [7], which is based on the idea of finding the minimum description length (MDL) of the objects' classes in the training data. The description consists of two parts: the model and the exceptional data. Induced tree classifies the training data completely so that there is no exceptional data. However, the model, i.e. the structure of the tree, may be unreasonably complicated for example due to noise. MDL has turned out to give a good and intuitive compromise between the model complexity and the exceptional data. The length of the description is measured as the number of symbols needed to encode it. This poses a problem, which limits the feasibility of the MDL principle: the most efficient encoding scheme should be available. MDL-SLIQ is an algorithm that makes some simplifications in measuring the coding lengths. It traverses the

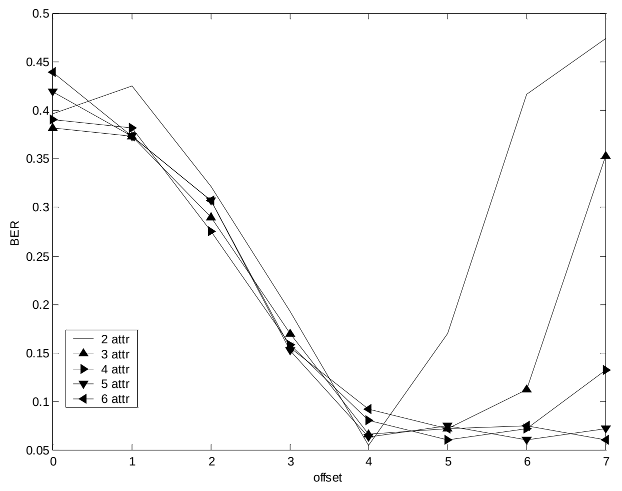


Figure 2: Classification tree (non-pruned) bit error rate as a function of *offset*. SNR = 5, average of 10 bursts was calculated.

maximum tree in bottom-up order and turns the current node into a leaf by cropping both child nodes away in case it provides a shorter description length. The majority class among the examples falling on a leaf will determine the prediction $\hat{a}$ attached to that leaf.

EXPERIMENTAL RESULTS

In this study we present simulation results for our tree classifier and compare them to the corresponding results achieved with a cascade-correlation trained multilayer perceptron neural network equalizer [8]. Tree induction and neural network learning are performed for each burst separately. We used a fixed channel impulse response $\boldsymbol{h}=[h_0\ h_1\ \dots\ h_4]=[0.5\ -0.3\ 0.6\ -0.7\ -0.8]^T$, and signal to noise ratio (SNR) was varied from 5 to 30 dB. The equalizer can be expressed as

$$\begin{aligned} \hat{a}_n &= f(\mathit{attribute}\ 1, \mathit{attribute}\ 2, \dots, \mathit{attribute}\ K) \\ &= f(y_{n+\mathit{offset}}, y_{n+\mathit{offset}-1}, \dots, y_{n+\mathit{offset}-K+1}) \end{aligned} \tag{4}$$

where $\hat{a}_n$ is an estimate of the symbol a_n and f is a nonlinear equalizing function, i.e. a classification tree or a neural network. In this representation *offset* stands for adjusting the indexing between y and a. Radio path delay is omitted so that a_n and y_n represent the same time instant. Therefore the delay of the signal is totally determined by the phase response of the channel. Positive *offset* means that the equalizer is actually noncausal, it is waiting also for future values of y to make a decision $\hat{a}_n$ for the present data bit a_n. It would usually be desirable to capture all the energy of the channel's response to a transmitted symbol, but in some applications increasing *offset* may have to be restricted in consequence of increasing decision delay. If the channel is very noisy, the importance of choosing the attributes is indisputable, as is clearly visible in Fig. 2. In case of only two attributes, *offset = 4* would be the only choice to achieve good performance. Using more attributes instead gives additional margin in choosing *offset*, i.e. an equalizer with more attributes is less sensitive to mismatched *offset*. The reason why a moderately long delay is desirable in this case is clarified by observing that the tail coefficients of ***h*** have the greatest magnitude.

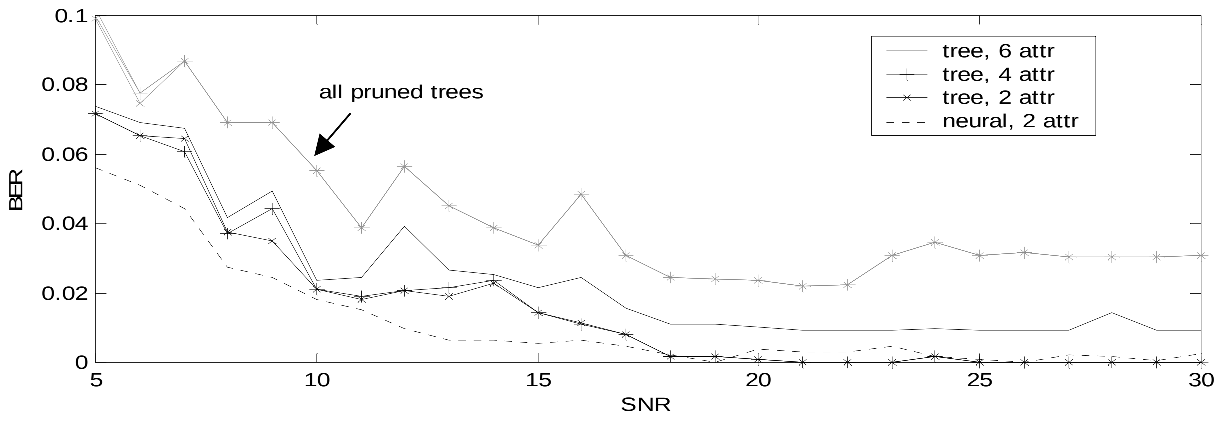
We are able to draw yet another conclusion by studying Fig. 2. Approximately the same performance can be achieved with any number of attributes upward of two. The inductive bias of a tree classifier is a preference for small trees over large trees. Therefore only few consecutive attributes are actually used in most cases, and the structure of the tree is less frequently dependent of the value of *K*. That feature strengthens the feasibility of classification trees, as overcomplex models are not very likely. In fact Fig. 3 a) indicates that it is questionable whether a pruning algorithm should be used at all. Pruning simplifies the model at the expense of accuracy. The utilization of pruning is better justified in applications where, in addition to noise corruption, the number of training examples is large.

Based on Fig. 2 we chose the optimum value *offset = 4* to be used in our overall performance comparison, which is given in Fig. 3. We can clearly see that pruning results in a considerable decline of classification performance regardless of SNR or the number of attributes used. The similarity of all curves suggests that structures of pruned trees are exceedingly simple and mostly alike. Let us concentrate on non-pruned trees for a while. Neural network equalizer performs slightly better in case of severe noise corruption, but when SNR exceeds 15 dB the differences are not significant anymore. On the other hand, the number of floating point operations used in learning is much smaller with tree classifiers. The extreme case of SNR = 5dB and two attributes gives a complexity ratio of 38 for the benefit of the tree classifier. Computational complexity of the neural network depends on the number of inputs (in our case 2), training epochs (50) and hidden units (max. 6).

CONCLUSIONS

In this paper, a classification approach to the equalization and detection process has been presented. We discussed how signal space interpretation, used for example in pattern analysis, can be used in our application. Classification trees are a competitive choice when simplicity and comprehensibility of classifier structure are desired. For future work, a few things are suggested. Firstly, information gain criterion does not fit perfectly for numerical attributes and noisy environments, as it does not consider distances between distinct examples at all. Furthermore, we have not investigated alternative pruning methods, it is possible that clear improvements could be achieved. Our prospective research will concentrate espe-

a)



b)

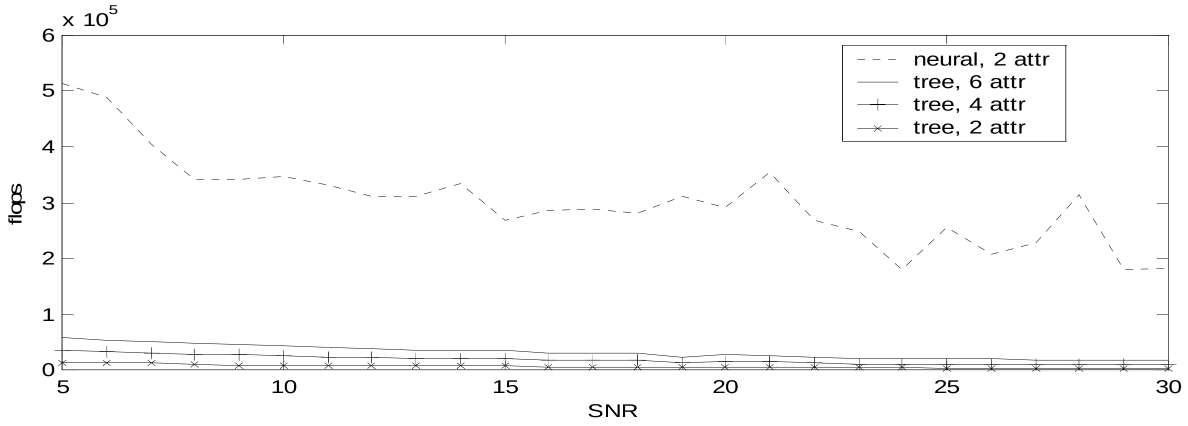


Figure 3: a) Bit error rate after classification. Black solid lines represent trees prior to pruning, gray lines represent pruned trees. BER of neural network equalizer is plotted with dashed line. All results are achieved as an average of ten bursts, using *offset = 4*. **b)** The number of floating point operations used in learning. Pruning did not affect the complexity noticeably.

cially on more realistic system model and the matters discussed above.

REFERENCES

[1] G. D. Forney Jr., "Maximum likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, May 1972, pp. 363-378.

[2] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 1995.

[3] S. U. H. Qureshi, "Adaptive equalization", *Proc. IEEE*, vol. 73, Sept. 1985, pp. 1349-1387.

[4] M. Mouly and M-B. Pautet, *The GSM System for Mobile Communications*. Palaiseau: Mouly & Pautet, 1992.

[5] S. Kuusisto, *Application of the PMDL Principle to the Induction of Classification Trees.* Dr.Tech. thesis, Tampere University of Technology, Publications 233, 1998.

[6] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[7] M. Mehta, R. Agrawal and J. Rissanen, "SLIQ: a fast scalable classifier for data mining," *Proc. of the 5th International Conference on Extending Database Technology*. Avignon, France, March 1996.

[8] A. Kantsila, M. Lehtokangas and J. Saarinen, "Adaptive equalization of binary data bursts with cascade-correlation trained multilayer perceptron networks," *Proc. of the 18th IASTED International Conference on Modelling, Identification and Control (MIC'99)*. Innsbruck, Austria, Feb. 1999, pp. 438-441.