# Decision Tree Design from a Communication Theory Standpoint

RODNEY M. GOODMAN, MEMBER, IEEE, AND PADHRAIC SMYTH

*Abstract*—**The design of efficient decision trees from labeled sample data is currently an important topic in several fields, such as pattern recognition and expert system design. A communication theory approach to decision tree design based on a *top-down* mutual information algorithm is presented. We state and prove that this algorithm is equivalent to a form of Shannon–Fano prefix coding and derive several fundamental bounds relating decision tree parameters. We then use these bounds, in conjunction with a rate-distortion interpretation of tree design, to explain several phenomena previously observed in practical decision tree design. We also propose a new termination rule for the algorithm, called the delta-entropy rule, which improves the robustness of the algorithm in the presence of noise, compared with existing methods. Simulation results are presented from which we show that the tree classifiers derived by our algorithm compare favorably to the single nearest neighbor classifier.**

## I. INTRODUCTION

HIERARCHICAL classifier design is a subject which has received considerable attention in recent literature. In designing any classifier, hierarchical or not, one seeks to classify some future samples of unknown class based on what one has inferred from labeled training samples. The training samples are described by feature vectors located in a multidimensional feature space. In addition, each sample has a label, i.e., the class to which it belongs. Classifier design can thus be viewed as a statistical inference procedure for partitioning the multidimensional feature space into class regions. Future samples are thus assigned to a particular class region, or classified, according to some function of its feature vector in this space. It is important to note that this problem is quite general and could equally well describe problems in speech recognition, image analysis, data compression, medical diagnosis, equipment maintenance strategies, etc.

The more traditional techniques of partitioning the feature space, as described in Duda and Hart [1], rely on extensive knowledge of the joint probability distributions between the classes and the feature vectors. In addition, they yield computationally complex classifiers where, typically, an unknown sample must be compared to each class to find the "most likely" classification. Estimation of the joint distributions when either the dimensionality of the feature space or the number of classes is very large requires impractically large training sets. Indeed, increasing the number of features while keeping the number of samples fixed can actually lead to a decrease in the accuracy of the classifier [2].

An alternative approach is to use a nonparametric hierarchical partitioning of the feature space. Clearly, as in any hierarchy, the order in which the hierarchy is implemented is of fundamental importance for any optimality criterion which might be used. This hierarchical approach can be traced back to Wald's original work on sequential statistical decision theory [3]. The idea of using the mutual information between the features and classes to select the best features has received considerable attention recently (cf. [7]–[20]), but in fact the idea was initially put forward in 1962 in a paper by Lewis [4]. Then in 1968, Fu [5] formulated Wald's ideas in terms of the classifier design problem with reference to Lewis' information criterion. More recently, there has been renewed interest in this approach, more commonly referred to as decision tree design (since a hierarchical classifier can be viewed as a decision tree). The primary reason for this renewed interest is the need to derive more efficient classifiers, particularly with respect to classification speed. More detailed rationale on the motivation for using hierarchical classifiers can be found, for example, in Kanal [6] and Swain and Hauska [7].

Some of the earlier work (e.g., Ganapathy and Rajaraman [8], Hartmann *et al.* [9]) deals with the conversion of decision tables to decision trees. However, decision table conversion in some respects is a subset of the more general problem of tree design from a table of arbitrary sample data, as outlined by Chou [10]. We shall deal with a particular approach to this more general problem of probabilistic decision tree design, namely, using the average mutual information between the features and the classes to design the tree. The basic principle of this approach revolves around choosing the "best" feature at any node in the tree (or, equivalently, at any stage in the sequential decision process), *conditioned* on which features were chosen previously and the outcomes of evaluating those features. From an intuitive viewpoint, any sequence of feature evaluations or path from the root node to a leaf (i.e., a terminal node) involves only those features which are most

Manuscript received March 23, 1987; revised January 27, 1988. This work was supported in part by Pacific Bell. This paper was presented in part at the 1986 IEEE International Symposium on Information Theory, Ann Arbor, MI, October 1986.

The authors are with the Department of Electrical Engineering 116-81, California Institute of Technology, Pasadena, CA 91125.

0018-9448/88/0900-0979$01.00 ©1988 IEEE

relevant toward the classification decision made at the leaf. Clearly, some trade-off must exist between the number of features evaluated and the accuracy of classification. Recently, Chou and Gray [11] have formulated this in terms of rate-distortion theory, where the classification error versus average depth trade-off can be directly related to the rate-distortion curve for a given problem. The overall effect one expects of using the average depth approach is for the average number of feature evaluations to be considerably reduced, resulting in advantages in time and cost. Among the successful applications of the mutual information approach to decision-tree design reported already are medical diagnosis [12], an expert system design tool [14], alpha-numeric character recognition [15], [16], Chinese character recognition [17], and the classification of chess endgames [18]. In addition, several successful applications of tree classifiers have been designed manually without using information-theoretic concepts, e.g., in aerial image analysis [19] and speech recognition [20].

Within the domain of the mutual information approach are several variations on the same principle, e.g., Breiman *et al.* have developed extensive algorithms based on pruning large or complete trees back to much smaller trees [12]. We shall deal with algorithms which are strictly *top-down* where the tree is essentially derived in a single iteration. In 1982, Hartmann *et al.* [9] proposed a general *top-down* mutual information algorithm to design decision trees from deterministic decision tables. Recently, this work has been extended [13] to *probabilistic* decision trees. Our perceived limitations of this algorithm are that it is restricted to deal only with independent tests and that it requires that all the probabilities are known *a priori*. In general, for practical tree-design problems, such as may occur in expert system design applications, such conditions are not satisfied.

We deal here with the more general problem, namely, tree design from a table of sample data using the *top-down* mutual information algorithm. The tests or features are *not* assumed to be independent, and the probabilities are estimated from the sample data. We relate the problem to communication theory using a noisy channel analogy. Specifically, we prove that the algorithm is directly equivalent to a form of Shannon–Fano prefix coding. This leads to considerable insight into the basic principle involved, from which we derive bounds relating the various tree performance parameters, such as the average tree depth and probability of misclassification. We analyze the problem of determining suitable termination rules for the algorithm (i.e., when to declare a tree node to be a leaf or terminal node) and propose a new termination rule called the delta-entropy rule to overcome the disadvantages of threshold-type rules. Finally, we present experimental results and interpret them in the light of the theoretical results given earlier. The primary conclusions are that the greedy algorithm (i.e., *top-down* tree design using mutual information) will work well on the average and that the termination rules must be chosen carefully if noise is present in the feature vectors.

| *Samples* | *Attributes* $A_1$ $A_2$ ..... $A_N$ | | | *Classification label* |
|---|---|---|---|---|
| 1 | 0 | 35 | ..... red | $c_4$ |
| 2 | 1 | 16 | blue | $c_2$ |
| 3 | 1 | 42 | green | $c_{11}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| M | 0 | 32 | ..... red | $c_4$ |

Fig. 1. Typical sample data table.

## II. The Mutual Information Algorithm for Tree Design

In the introduction we developed an outline of the basic algorithm. Let us now be somewhat more specific. Consider that one is given a table of data as in Fig. 1. The table consists of $M$ samples drawn from the population at large. Each sample is described in terms of $N$ feature values (or *attribute* values) and a class label. The class labels are letters from the alphabet of the discrete random variable $\boldsymbol{C}$. For example, the samples could be plants described in terms of height, number of flowers, etc., or medical case histories in terms of presence or absence of certain symptoms. It is important to realize that the table is probabilistic rather than deterministic, i.e., "overlap" may exist in the class-feature conditional probability distributions.

Note that the problem of feature selection is not dealt with here; rather, we take what features are given and do the best we can. Feature selection is a different problem but nevertheless directly related. Indeed, if all known features are given then the tree-design algorithm will *automatically* select the most relevant features and ignore the irrelevant ones. This is an additional advantage of the mutual information tree-design approach over conventional classifiers since it yields valuable information on the relative importance of the various features. In applications such as medical diagnosis this can be quite useful [12]. We assume that the sample size is sufficiently large to yield reliable estimates of the class distribution conditioned on the values of the $N$ features. Indeed, it can be shown [21] that for a given confidence level the required number of samples is not too large from a practical point of view. Implicitly, we have also assumed that the feature values have been quantized—we will return to this point later.

Let $\boldsymbol{C}$ be a discrete random variable, henceforth referred to as the class variable. The finite set of class labels $\{c_1, c_2, \cdots, c_K\}$ comprises the letters of the alphabet of $\boldsymbol{C}$. The probability that $\boldsymbol{C}$ assumes the value $c_j$ is denoted by $p(\boldsymbol{C} = c_j)$ and $\sum_{j=1}^{K} p(\boldsymbol{C} = c_j) = 1$. We will adopt the convention that $p(c_j) = p(\boldsymbol{C} = c_j)$.

```
tree()
    if (stack is not empty)
        { node = pop_node()
          leaf_flag = leaf (node)
          if (leaf_flag = 1)
              { increment leaf_list
                tree()                }
          else
              { A i = max_inf._attribute(node)
                for ( k = 1 to n i )
                    { sort_table (a k)
                      push_node( ) }
              }
        }
    else return ( )
end
```

Fig. 2. Pseudocode description of algorithm.

In addition, we have $N$ features or attributes. Each feature $A_i$, $1 \le i \le N$, is also defined as a discrete random variable, i.e., the $i$th feature or variable has an alphabet $\{a_1^i, a_2^i, \cdots, a_{n_i}^i\}$, where $n_i$ is the cardinality of the alphabet of $A_i$. The probability of the event $A_i = a_j^i$, $1 \le j \le n_i$ is denoted as $p(A_i = a_j^i)$, where $\sum_{j=1}^{n_i} p(A_i = a_j^i) = 1$, $1 \le i \le N$.

Now we consider the algorithm itself. Essentially, it just deals with one tree node at a time. The initial node (root node) consists of the original table of data samples, i.e., unconditioned on any feature values. Subsequent nodes result from evaluating certain features and obtaining subtables conditioned on the outcome of all prior evaluations. The algorithm continues to process nodes until no candidate nodes remain, i.e., the tree has been grown and all leaves and internal nodes defined. The algorithm may be implemented recursively using a stack to store unprocessed nodes. New nodes which are not leaves are pushed onto the stack and "popped" off later to either yield more new nodes (child nodes) or be declared a leaf. The algorithm is defined in pseudocode in Fig. 2.

Apart from the bookkeeping aspects, two functions in the algorithm remain to be defined, namely, the "leaf-or-not" and "max-information-feature" functions. It is these two functions which characterize any *top-down* tree derivation algorithm, i.e.,

a) determine if the node is a leaf;
b) if not, determine the feature which yields the maximum information at that node.

We shall deal with the first problem which concerns finding appropriate termination rules later. Initially, we will focus on the second problem, namely, which feature yields the most information. The criterion can be stated quite simply, but we shall be more interested in what implications this approach has for the tree as an efficient classifier. We can state the criterion in terms of our previously defined notation. Choose the feature $A_k$ such that

$$I(C; A_k) \ge I(C; A_i), \qquad 1 \le i \le N,\ i \ne k$$

or, equivalently,

$$H(C|A_k) \le H(C|A_i), \qquad 1 \le i \le N,\ i \ne k \qquad (1)$$

where $H(\cdot)$ and $I(\cdot)$ are the well-known entropy and average mutual information functions, respectively. This is essentially the same as the criterion originally proposed by Lewis in 1962 [4]. In his paper Lewis justifies theoretically why the mutual information criterion is appropriate for feature selection—the interested reader should refer to the original.

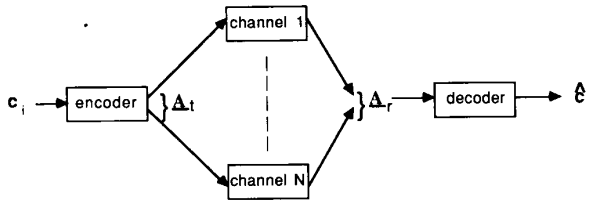


Fig. 3. Communications channel model of feature measurement.

We now introduce an interesting analogy with a communications problem. The transmitted information is $c_i$, the unknown class (from the receiver's viewpoint), a member of what one might term the class alphabet. What is being transmitted is essentially a coded version of this "letter" $c_i$, namely, $\bar{A}_t$, the transmitted feature vector. This feature vector (or message sequence) contains $N$ components, and each component can assume $n_i$ different elements in each alphabet. Consider that the features are transmitted over $N$ different discrete memoryless noisy channels, one for each feature component and its alphabet. Fig. 3 shows the overall picture. For example, feature $a_1$ could be binary values ($n_1 = 2$), and so channel 1 could be a binary symmetric channel. These $N$ noisy channels are analogous to the feature measurement process and essentially represent a model for the combined effects of overlap in the joint class-feature distributions, transducer noise, quantization noise, etc.

The receiver's problem is to compute $\hat{c}$ as a best estimate for whichever $c_i$ was transmitted. This is equivalent to decoding the received feature vector $\bar{A}_r$.

Clearly then, having a set of training samples available in the classification problem is analogous to having a number of test transmissions with side information available, namely, the true value of $\hat{c}$. The problem is to design a decoder which is optimal or near optimal in some sense, given what one has inferred from the test transmissions. Traditionally, the approach is to use all of the available channels (i.e., all of the features) in the decoding algorithm. Perhaps, however, some channels are costly to operate, just like some features are difficult to measure, e.g., if one of the channels is an expensive satellite link. The hierarchical approach here is to devise a decoding algorithm which uses the channels sequentially and terminates

with an estimate of $\hat{c}$, using on average somewhat fewer than $N$ channels. The mutual information extension of this is simply to choose sequentially the channels which yield the maximum average mutual information between $\bar{A}_r$ and $c_i$.

## III. Tree Derivation as a Form of Prefix Coding

As outlined in the previous section, the tree algorithm can be reduced to a recursive procedure which only deals with nodes. Each node in the designed tree has an associated test or feature evaluation. In our notation this feature is a random variable. We adopt the convention (for notational convenience) of using $\text{node}_j$ to represent the feature variable associated with the $j$th node, where $j$ is an index over the internal nodes. In other words, wherever the $\text{node}_j$ symbol is used it represents the random variable associated with that node. Hence we can think of the node itself as a random variable whose outcomes are members of the associated feature alphabet. In a similar manner, we define the discrete random variable $\boldsymbol{T}$ to be a function of the internal nodes. Henceforth we refer to $\boldsymbol{T}$ as the tree. The alphabet of $\boldsymbol{T}$ consists of all the possible paths (or sequences of feature values) through the designed tree. Since these paths are disjoint and the sum of their probabilities is 1, $\boldsymbol{T}$ is indeed a random variable.

We define the average mutual information $I(\boldsymbol{C};\boldsymbol{T})$ which the tree yields about the classes as

$$I(\boldsymbol{C};\boldsymbol{T}) = \sum_{j \in S} p(\text{node}_j) I(\boldsymbol{C};\text{node}_j) \tag{2}$$

where $S$ is the set of internal nodes, $p(\text{node}_j)$ is the probability that one traverses a particular internal node ($j$ is an arbitrary index) in the tree, and $I(\boldsymbol{C};\text{node}_j)$ is the information which the node yields about $\boldsymbol{C}$, i.e., by traversing a node one evaluates a feature and descends a branch to a child node conditioned on the outcome of the feature evaluation. The average mutual information gained is the average information one receives about the classes when one evaluates this feature. Note that to calculate $I(\boldsymbol{C};\boldsymbol{T})$ in (2) we need a specific designed tree; i.e., $I(\boldsymbol{C};\boldsymbol{T})$ depends on the sample data and the algorithm used. A natural question to ask is "what exactly is $I(\boldsymbol{C};\text{node}_j)$?"

*Theorem 1:* The average information gained by evaluating a feature at a given node is simply the entropy of the probabilities of the branches leaving that node:

$$I(\boldsymbol{C};\text{node}_j) = H_m(q_1, q_2, \cdots, q_m) = \sum_{i=1}^{m} q_i \log\left(\frac{1}{q_i}\right) \tag{3}$$

where $m$ is the number of branches at the node and $q_i$ $(1 \le i \le m)$ is the probability of descending the $i$th branch.

*Proof:* See Appendix I for details.

The result can be worded as follows. The information we gain from traversing a node in the tree is simply equal to the $m$-ary entropy of the branch probabilities emanating from that node. As a numerical example, if $m = 2$ and $q_1 = 0.4$ at some $\text{node}_j$, then by Theorem 1, $I(\boldsymbol{C};\text{node}_j) = H_2(0.4) = 0.971$ bits. We can state the following corollary.

*Corollary to Theorem 1:* The top-down or "greedy" algorithm for designing trees using mutual information is directly equivalent to prefix coding of a certain type, namely, a form of Shannon–Fano prefix coding.

*Proof:* From Theorem 1, the tree-design algorithm tries to maximize the quantity $H(p_1, \cdots, p_m)$. This is equivalent to determining the $m$-ary partition of the component $p_i$ such that the $p_i$ are as equal as possible. This is the same criterion as used in a form of Shannon–Fano prefix coding [22], and since trees are directly equivalent to prefix codes [23], the tree-design algorithm and the prefix coding algorithm are, in fact, the same. This is sufficient to prove the corollary.

Note, however, that the tree-design is really equivalent to "constrained" prefix coding in a practical sense since there is no guarantee that the features exist to perform the appropriate partitions. With actual prefix coding no such constraints exist.

An immediate consequence of this result is the fact that we have proven that the top down tree-design algorithm using mutual information is necessarily suboptimal. This is because of the equivalence to prefix coding where the optimal code derivation algorithm is that of Huffman [24]. We are, of course, interpreting optimal here in the sense of minimum average tree length which is consistent with our overall criterion of optimality when all feature costs are the same.

Therefore, if the *top-down* "greedy" algorithm is suboptimal, should we abandon it and search for the optimal one? Most probably not, for a variety of reasons, not least the fact that optimal tree derivation has been shown to be *NP*-complete [25] and hence would be computationally intractable as the number of classes and/or features increases. In addition, it is well-known that Shannon–Fano prefix coding yields near-optimal length codes in practice. One suspects then that this simple algorithm may be the "best" from a practical point of view. Rather than viewing the proof of suboptimality of the algorithm as a negative result, one should interpret its equivalence to Shannon–Fano coding as a positive argument in its favor. The only weakness in the argument is the lack of quantitative results available on how "near-optimal" this prefix coding scheme really is—although it is known to work well in practice one would like to quantify this. Section IV will establish some results in this regard.

Theorem 1 was derived on the unrealistic assumption of zero noise. The reason for doing this was to clarify the basic equivalence to prefix coding. Indeed, the following theorem includes the result of theorem 1 as a special case,

but we have chosen to separate the two results for the purposes of clarity.

*Theorem 2:* The following holds:

$$I(\boldsymbol{C};\text{node}_j) = H_m(q_1, q_2, \cdots, q_m) - H_m\left(1-\epsilon, \frac{\epsilon}{m-1}, \cdots, \frac{\epsilon}{m-1}\right) = \sum_{i=1}^{m} q_i \log\left(\frac{1}{q_i}\right) - (1-\epsilon)\log\left(\frac{1}{1-\epsilon}\right) - \epsilon \log\left(\frac{m-1}{\epsilon}\right) \quad (4)$$

where $m$ is the number of branches at the node, $q_i$ ($1 \le i \le m$) is the probability of descending the $i$th branch, as before. Now $\epsilon$ is the probability that one will not descend the "correct" branch and $\epsilon/(m-1)$ is the probability that one descends any of the other $m-1$ branches.

For a proof of this result using a noisy channel analogy see Appendix I. From the proof we note that this is simply the information equation for a discrete memoryless channel so that the noise term can be replaced by a general equivocation term $H(\boldsymbol{Q}|\boldsymbol{C})$ for any noise characteristics, symmetric or not, where $\boldsymbol{Q}$ is the $m$-ary partition random variable as defined in the Appendix. Let us say that $m = 2$ and $q_1 = 0.4$ as before, and we let $\epsilon = 0.1$. From (4) we then find that $I(\boldsymbol{C};\text{node}_j) = H_2(0.4) - H_2(0.1) = 0.502$ bits. In other words, the introduction of a noise level of $\epsilon = 0.1$ resulted in a loss of information of 0.469 bits.

Here we see that the noise places a fundamental limit on the amount of information a node can yield, i.e.,

$$I(\boldsymbol{C};\text{node}_j) \le \log_2(m) - (1-\epsilon)\log(1/(1-\epsilon)) - \epsilon \log(m-1/(\epsilon)).$$

From the original equation it might seem that $I(\boldsymbol{C};\text{node}_j)$ could be negative if the noise were large enough. Of course, this does not happen since the $q_i$ themselves are noise-dependent and with complete randomness they must in fact be equal giving the $\log_2(m)$ term for the first expression on the right side. We shall later see that this limit has direct implications for using threshold-type termination rules.

## IV. Bounds on the Average Mutual Information Available at a Node

We have seen that the average information to be gained from a node is equal to $H_m(q_1, q_2, \cdots, q_m)$, where $q_i$ is the probability of descending to the $i$th child given that one is at the parent node. Clearly, this entropy term attains a maximum when all the $q_i$ are equal to $1/m$. On average, however, given an arbitrary class probability distribution at the node, how well can one do in terms of maximizing the information?

Consider a binary node, i.e., a node where one wishes to partition the members of the class alphabet into two disjoint subsets. Or to look at it from another angle, how much information can one get by asking a purely binary question? Let us define $p_{\max}$ as the maximum probability component in the discrete distribution of the class random variable $\boldsymbol{C}$. Then we have the following theorem.

*Theorem 3:* Given a discrete random variable $\boldsymbol{C}$ with $K$ possible outcomes, one can always define a partition of the outcomes of $\boldsymbol{C}$ into two disjoint subsets such that

$$I(\boldsymbol{C};\boldsymbol{Q}) \ge H_2\left(\max\left\{p_{\max}, \frac{1}{3}\right\}\right) \quad (5)$$

where $I(\boldsymbol{C};\boldsymbol{Q})$ is the average mutual information between the class variable $\boldsymbol{C}$ and the optimal partition variable $\boldsymbol{Q}$. The optimal partition variable $\boldsymbol{Q}$ is, in turn, defined as the binary random variable whose two possible outcomes consist of the optimal partitioned subsets of the alphabet of $\boldsymbol{C}$. The optimal split will be the one for which the probabilities of the components of $\boldsymbol{Q}$ are closest to 0.5.

*Proof:* See Appendix II.

We plot this bound as a function of $p_{\max}$ in Fig. 4. This leads to the following corollary.

*Corollary to Theorem 3:* The following holds:

$$\text{if } p_{\max} < \frac{2}{3}, \qquad \text{then } I(\boldsymbol{C};\boldsymbol{Q}) > H_2\left(\frac{1}{3}\right) = 0.918 \text{ bits.} \quad (6)$$

*Proof:* Follows directly from Theorem 3.

Theorem 2 and its corollary yield a surprising result. For example, the corollary states that if the maximum probability component is less than 2/3, then one can always define a partition or question which yields 0.918 bits of information. This is quite large when one considers that the upper bound is, of course, 1 bit so that even under the most adverse conditions we lose only 0.082 bits. For $p_{\max} < p \le 2/3$ the loss of information is even smaller. One can already begin to see that top-down node splitting may well be near optimal under certain conditions. Later we shall use this theorem to show why Shannon–Fano prefix coding and more relevantly, *top-down* tree design, almost always yield near-optimal results.

Consider first though the question of how restrictive the assumption that $p_{\max} < p$ may be, i.e., how likely it is that an arbitrary distribution will have $p_{\max}$ less than some value $p$, say 2/3 or 9/10 or whatever. Assume that no constraints have been put on the distribution in a Maxwell–Boltzmann statistical type of argument, i.e., every possible distribution is equally likely. It can be shown [21] using a combinatorial occupancy technique that

$$\text{pr}\{p_{\max} > p\} = k(1-p)^{k-1} \quad (7)$$

where $0 < p_{\max} \le 1$, $0.5 < p \le 1$, and $k$ is the number of components of the distribution, e.g., $k = K$ for the class variable $\boldsymbol{C}$. Consider a numerical example where $k = 10$ and $p = 0.9$. From (7) we get that $\text{pr}\{p_{\max} > p\} = 10 \cdot (1 - 0.9)^{10-1} = 10^{-8}$.

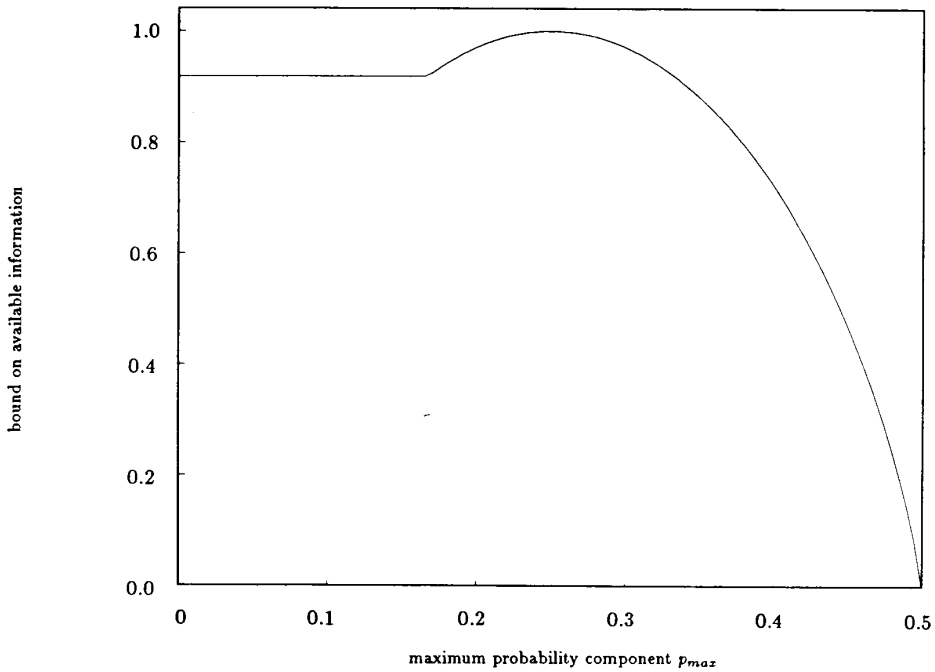


Fig. 4. Lower bound on maximum average information available by asking binary question, as a function of $p_{\max}$.

Clearly, the probability depends primarily on the number of classes. As the number of classes $k$ increases, the probability tends to zero. Of course, just as in solid-state physics the assumption that all the "states" or distributions are equally likely may be an over simplification. Nevertheless, the result is a positive indication that as the number of classes increases, "almost all" distributions have no components greater than some $p$, i.e., for any arbitrary $\delta$ there exists a $p$, $0.5 \le p < 1$, such that

$$\text{pr}\{ p_{\max} > p \} = k(1-p)^{k-1} < \delta. \tag{8}$$

Returning to the main topic of interest, we now extend the result of Theorem 3 to account for any noise which may be present at the node. Consider a noise level of $\epsilon$ at the node as previously defined.

*Theorem 3 Extended:* The following holds:

$$I(\boldsymbol{C};\boldsymbol{Q}) \ge H_2(p + \epsilon(1-2p)) - H_2(\epsilon) \tag{9}$$

where $p = \max\{ p_{\max}, 1/3 \}$.

*Proof:* See Appendix II.

The consequences of this result can be seen in Figs. 5 and 6. Fig. 5 relates to $p_{\max} = 0.9$, while Fig. 6 is for $p_{\max} = 2/3$. The upper bound in each graph is $1 - H_2(\epsilon)$, the maximum amount of information available for a noise level of $\epsilon$. The lower bound is the minimum amount of information which can be achieved by the optimal partition *given* that the maximum probability component is less than some $p_{\max}$; i.e., constraining the maximum probability to be less than $2/3$ is stronger than constraining it to be less than 0.9, which explains why the bound is much tighter in the former case. The noise level $\epsilon$ is only allowed to 0.5 since beyond that point the curve is symmetric. This is consistent with the idea of communication channel type noise. The extended result gives a direct quantitative measure of the decrease in node information due to noise.

## V. Information Theoretic Bounds on the Tree Performance

We now pose the question of how the tree depth, tree information, and the probability of misclassification relate to each other. Clearly, the latter two parameters can be compared by Fano's inequality (cf. [16] and originally, [22]), and the tree depth will be introduced using results derived earlier. First it is informative to review what type of relationships one might reasonably expect to hold. Consider a situation where there is some noise $\epsilon$, and view this in terms of Chou and Gray's rate-distortion model [11]. The rate is equivalent to the average tree depth while the distortion is the probability of misclassification. Fig. 7 depicts the type of distortion-rate curve one might expect based on Wolf and Ziv's original model [26] for this type of problem, where the original inputs (the original class information) are distorted (represented by an imperfect set of features) prior to being source coded (decision tree) and transmitted (measured by an observer) before decoding (classification at a leaf).

Let us define $\bar{d}$ to be the average tree depth and $P_e$ to be the average probability of misclassification for the tree. Also, note that $H(\boldsymbol{C})$ is the entropy of the class distribution (before using the tree), $I(\boldsymbol{C};\boldsymbol{T})$ is the average mutual information which the tree yields about the classes, and $H(\boldsymbol{C}|\boldsymbol{T})$ is the average remaining uncertainty about the classes having used the tree.

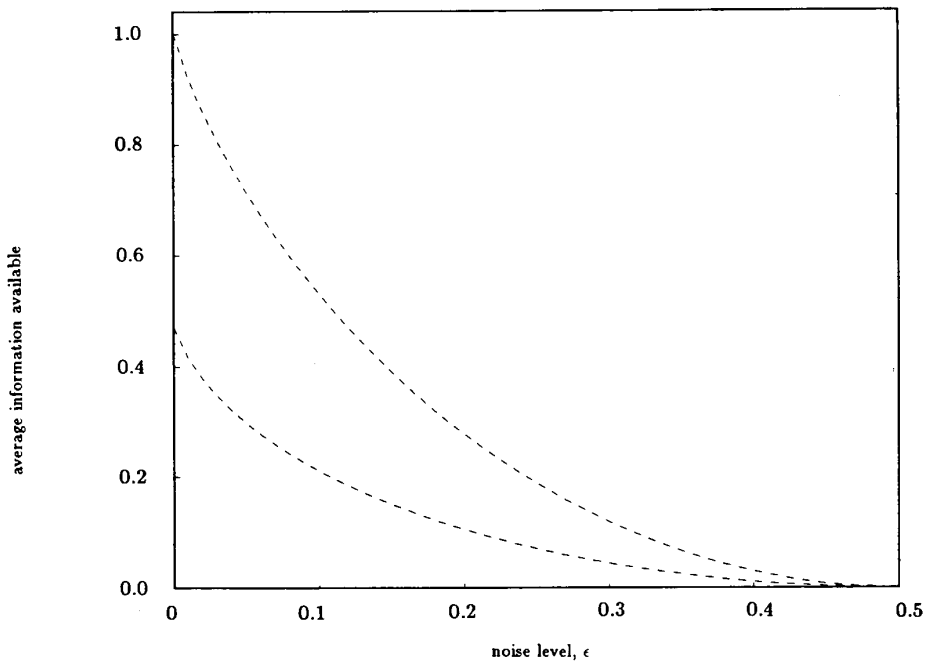


Fig. 5. Upper and lower bounds on decrease in available information as functions of $\epsilon$: $p_{\max} = 9/10$.

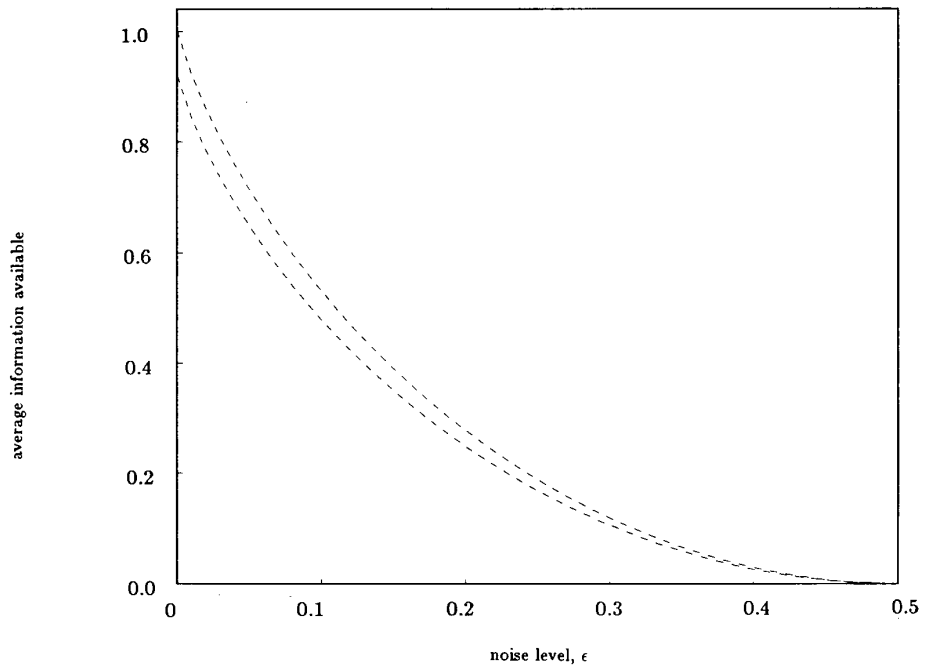


Fig. 6. Upper and lower bounds on decrease in available information as functions of $\epsilon$: $p_{\max} = 2/3$.

Some general comments are in order at this point. The hierarchical classification approach is based on the principle of finding an "operating point" slightly above the distortion-rate curve itself at a rate where the distortion-rate curve itself is slightly above the Bayes misclassification rate. This involves the following two assumptions.

1) There is substantial redundancy in the feature set with respect to the class distribution, i.e., $H(\boldsymbol{C})$ is very much less than $N$, the number of features. If this is true, one expects the distortion-rate curve to approach the Bayes misclassification rate for $\bar{d}$ somewhere above $H(\boldsymbol{C})$ but still "far away" from $N$.

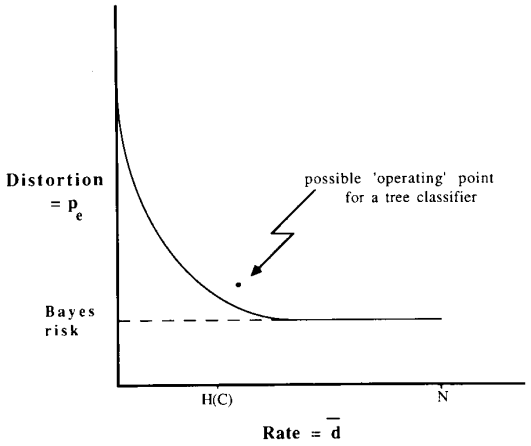


Fig. 7. Typical distortion-rate characteristic for classification problem.

2) In addition, one assumes that one can approach the distortion-rate limit by actually designing a near-optimum hierarchical classifier.

Essentially, assumption 1 is problem-dependent and can be taken to hold in a wide variety of well-known classification tasks, such as image classification, where a wealth of features exist many of which contain similar information; e.g., Conners and Harlow [27] have shown that many measures used in image processing to discriminate between textures contain the same information. First we bound the average tree depth. The lower bounds are quite general and involve no assumptions whatsoever while the upper bounds depend on the type of termination rule used in the tree-design algorithm, among other factors.

*Theorem 4:* For a binary tree with average depth $\bar{d}$ designed using the *top-down* mutual information algorithm

$$\text{a)} \qquad \bar{d} \geq \frac{H(\boldsymbol{C}) - H(\boldsymbol{C}|\boldsymbol{T})}{1 - H_2(\epsilon)} \tag{10}$$

This is true in general, while

$$\text{b)} \qquad \bar{d} \leq \frac{H(\boldsymbol{C}) - H(\boldsymbol{C}|\boldsymbol{T})}{H_2(p + \epsilon(1-2p)) - H_2(\epsilon))} \tag{11}$$

where a termination rule of the form "stop splitting when the maximum probability component at a node is greater than $p$" is used and the assumption is made that the optimal node splits can be defined at each internal node. For a proof of this result see Appendix III.

*Example:* Consider a uniformly distributed class distribution with 32 classes; the noise level $\epsilon$ is zero, and we use the threshold termination rule of $p_{\max} > 2/3$. Making the further assumption that the appropriate node splits can then be defined,

$$I(\boldsymbol{C};\boldsymbol{T}) \leq \bar{d} \leq \frac{I(\boldsymbol{C};\boldsymbol{T})}{0.918} = 1.09 I(\boldsymbol{C};\boldsymbol{T}) \tag{12}$$

$$\text{but } I(\boldsymbol{C};\boldsymbol{T}) = H(\boldsymbol{C}) = 5.0, \qquad \text{since } \epsilon = 0$$

$$\text{therefore } 5.0 \leq \bar{d} \leq 5.45.$$

The assumptions are certainly satisfied if one were prefix-coding the classes. Hence the tightness of the bound explains why Shannon–Fano coding "almost always" seems to work as well as it does; i.e., the average length of the code is lower bounded by $H(\boldsymbol{C})$ as always and upper bounded by $1.09H(\boldsymbol{C})$, provided that $p_{\max} < 2/3$ at each node.

The lower bound on $\bar{d}$ is fixed for *any* tree classifier so that the closeness of the bounds on $\bar{d}$ in the previous example is indeed a quantitative affirmation that the top-down tree-design algorithm does provide near-optimal trees. $H(\boldsymbol{C}|\boldsymbol{T})$ can be related to $P_e$ as follows:

$$H_2(P_e) + P_e \log(K-1) \geq H(\boldsymbol{C}|\boldsymbol{T}) \geq 2P_e \tag{13}$$

where the upper bound is due to Fano and the lower bound is derived in [28] as well as other sources. $K$ is the number of classes. Consequently, one can derive lower and upper bounds on $P_e$ in terms of $\bar{d}$, $H(\boldsymbol{C})$, and $\epsilon$, using (13) and (11):

$$H_2(P_e) + P_e \log(K-1) \geq H(\boldsymbol{C}) - \bar{d}(1 - H_2(\epsilon)) \tag{14}$$

$$P_e \leq \frac{1}{2}\left(H(\boldsymbol{C}) - \bar{d}(H_2(p + \epsilon(1-2p)) - H_2(\epsilon))\right). \tag{15}$$

As a numerical example of the upper bound (15), let $p = 2/3$ and $\epsilon = 0.1$. Let us say that the variable $\boldsymbol{C}$ is uniformly distributed and has an alphabet size $K$ of 8 so that $H(\boldsymbol{C}) = 3.0$. We then get that $P_e \leq 2.5 - 0.240\bar{d}$.

These bounds are not particularly tight in many instances due to the well-known fact that the general bounds relating error probability and uncertainty are loose. In addition, note that $H(\boldsymbol{C}|\boldsymbol{T})$ is that value which one has estimated from the available training samples. In this sense it is directly analogous to the resubstitution estimate for the misclassification rate in conventional classifier design problems. Given the nature of the algorithm, i.e. minimizing the remaining uncertainty, this estimate $H(\boldsymbol{C}|\boldsymbol{T})$ may be slightly lower than the true value. It follows then that the bounds just derived are actually for the estimated value of $H(\boldsymbol{C}|\boldsymbol{T})$ and the estimated misclassification rate (or the resubstitution estimate) so that the true probability of misclassification may be higher. Previous bounds derived in the literature [16] based on Fano's inequality have not emphasized this point. Unfortunately, no general relationship exists between the resubstitution estimate and the true misclassification rate, but in practice it has been found that if the sample size is sufficiently large then the true value of $P_e$ is not much greater than the resubstitution estimate [12].

It might seem that the upper bound on $P_e$ indicates that by arbitrarily increasing $\bar{d}$ one can reduce $P_e$ to zero. However, the distortion-rate model tells us that this cannot be true, or equivalently, no classifier can reduce $P_e$ below the Bayes misclassification rate. The anomaly is resolved by remembering that the upper bound depends on the assumptions made in deriving Theorem 4; in particular, it is obvious that one cannot keep asking questions *ad infinitum* and continue to receive much information. The lower bound is, however, completely general. Since $H_2(P_e) \leq 1$,

one can write the bound in a looser but more informative manner as follows:

$$P_e \geq \frac{H(C) - \bar{d}(1 - H_2(\epsilon)) - 1}{\log(K-1)} \tag{16}$$

e.g., for $\epsilon = 0.1$, $K = 8$, and $H(C) = 3.0$ as before, we get $P_e \leq 0.712 - 0.189\bar{d}$. This is, in fact, a weak lower bound on the distortion-rate characteristic. It is interesting that the bound is a linear function of the average depth of the tree, which confirms experimental results that increasing the average depth (given that the $1 - H_2(\epsilon)$ term is not zero, i.e., $\epsilon > 0$) will reduce the misclassification rate at least until the Bayes misclassification rate is approached. The increase in tree depth is necessary to exploit the inherent redundancy in the features. This is completely consistent with our communications analogy where the rate must be increased to reduce distortion.

## VI. Termination Rules

As stated earlier, one of the two fundamental criteria for any top-down tree design algorithm is its leaf termination criterion, i.e., the criterion used to decide whether a node should be a leaf or not. Before looking at the details of this problem it is worth referring briefly to the general distortion-rate curve of Fig. 7 once again. The distortion-rate curve is by definition a lower bound to the performance of any hierarchical classifier designed for that particular problem. It is reasonable to expect that tree classifiers will exhibit similar characteristics to the bound, i.e., as $P_e$, the probability of misclassification, is reduced and brought closer and closer to $p_{\min}$ (the Bayes misclassification rate), $\bar{d}$ must be increased accordingly.

The different types of termination rules essentially determine the tree's operating point, somewhere above the distortion-rate bound. Ideally, one would like to have a cost function $f(\bar{d}, P_e)$ for which the optimal tree would yield the minimum value of this function. Unfortunately, finding optimal trees has been shown to be an *NP*-complete problem [25]. Even finding near-optimal trees subject to a general cost function constraint is very difficult since

1) each particular problem may have its own cost function $f(\bar{d}, P_e)$;
2) the noise level $\epsilon$ is assumed to be unknown.

It is clear from what we have derived earlier that some algorithms will not work well in the presence of noise. Placing an upper bound on $P_e$ and continuing to split nodes until the resubstitution estimate is below this bound is not a good idea since, given a certain amount of noise (which we assume we do not know much about) and for a given problem, a fundamental lower limit exists on the misclassification rate as represented by the assymptotic limit of the distortion-rate curve as the rate approaches $N$, the number of features. Therefore if $P_e$ is chosen to be less than $p_{\min}$, even a complete tree (i.e., all features evaluated on the path to each leaf) may not satisfy the threshold condition. More generally, this approach is liable to yield very large values for $\bar{d}$ as $P_e$ approaches $p_{\min}$.

Similarly, one could choose to continue splitting until the total average information $I(\boldsymbol{C}; \boldsymbol{T})$ from the tree has exceeded a certain value or, equivalently, $H(\boldsymbol{C}|\boldsymbol{T})$ has decreased below a certain value. However, this is really equivalent to placing a threshold on $P_e$ since we have already seen that $H(\boldsymbol{C}|\boldsymbol{T})$ is bounded above and below by linear functions of $p_{\min}$. In other words, as well as there being a minimum misclassification rate $p_{\min}$ for a given problem with a certain amount of noise, there is a corresponding $H(\boldsymbol{C}|\boldsymbol{T})_{\min}$ where

$$2p_{\min} \leq H(\boldsymbol{C}|\boldsymbol{T})_{\min} \leq 1 + p_{\min}\log(K-1). \tag{17}$$

Therefore, constructing thresholds on $H(\boldsymbol{C}|\boldsymbol{T})$ or $I(\boldsymbol{C}; \boldsymbol{T})$ will not yield satisfactory results in the presence of noise for the same reason that constructing thresholds on $P_e$ will not work in the same situation.

Yet another approach which potentially looks somewhat more promising is to stop splitting nodes whenever $I(\boldsymbol{C}; \boldsymbol{Q})$ falls below a certain value, i.e., determine $I(\boldsymbol{C}; \text{node}_j)$ at $\text{node}_j$ for the best split, and if it falls below some threshold parameter $t$, then declare the node to be a leaf. The various results derived earlier in terms of $I(\boldsymbol{C}; \text{node}_j)$ and $\epsilon$ tell us that in the presence of noise, if $t$ is too large, one will get a lot of splitting and hence very large trees, while if $t$ is too small, very little splitting will occur because of the $H_2(\epsilon)$ term. The resulting tree will have $\bar{d}$ less than $H(\boldsymbol{C})$ with a resultant high probability of misclassification as can be inferred from the distortion-rate characteristic. In practice, this very phenomenon has been observed by other authors [12], [29] as a result of experimentation with designing trees for a particular problem and increasing the value of the noise level $\epsilon$. In the simulation results presented in Section VII we do not include results for trees designed using threshold rules since in the presence of noise such rules were found to be practically unworkable.

We have seen so far that threshold rules by themselves are not sufficient to form noise-independent termination rules. In addition, they have the added drawback of requiring some external intervention to supply the threshold levels, which is undesirable in the overall context of *automated* tree design.

Consider the basic problem again. One has a class probability distribution defined at a node, conditioned on the outcomes of feature (node) evaluations along the path from the root to the current node. The question is whether it is worth splitting this node or declaring it to be a leaf, i.e., making a classification decision at that point or evaluating another feature. Form the null hypothesis that the node is in fact worth splitting, i.e., the hypothesis is "another feature should be evaluated before making a classification decision." Potentially, many ways exist in which to test this hypothesis. The fixed threshold rules we have already considered are tests which are insensitive to any noise which may be present. An approach used by Quinlan [29] is to use the chi-square test to test for the independence of the candidate features (for splitting) and

the classes. The reasoning is that only the features which are rejected with a high degree of confidence are subsequently considered for splitting and so the effect of noise is minimized. However, as the number of classes and features increases, the number of samples at the node required to use the chi-square test becomes large. Since the test is for leaf nodes, it is precisely in the situation where nodes are deep in the tree and the number of samples at the node is relatively small that its use is critical to the success of the algorithm. Therefore, for problems with many classes and features this test may be insufficient as a termination criterion. Note, however, that for a two-class problem this rule was found to be extremely useful [29].

Given the limitations of the aforementioned rules we now propose a new termination test called the delta-entropy rule. No external parameters are required, and no constraints as to the number of samples are required for it to work. In addition, it is easy to compute and intuitively appealing from an information-theoretic viewpoint.

*The Delta-Entropy Rule:* If

$$H'(\boldsymbol{C}') \le H'(\boldsymbol{C}) \quad \text{and} \quad p_{\max} \ge 0.5, \tag{18}$$

then continue to split the node, where $\boldsymbol{C}$ is the original class random variable at the node, $\boldsymbol{C}'$ is the normalized class distribution if $p_{\max}$ is deleted from the component set, and $H'(\boldsymbol{P})$ is the usual entropy function divided by the logarithm to the base 2 of the number of nonzero probability components of $\boldsymbol{P}$.

The rationale is as follows. The hypothesis that the node is not a leaf node is equivalent to the hypothesis that apart from the class component corresponding to $p_{\max}$ other class components are present at the node which are not due to noise alone. If $H'(\boldsymbol{C}') \le H'(\boldsymbol{C})$, then this is evidence in favor of rejecting this hypothesis. In the absence of any other evidence the only alternative is to declare the node to be a leaf. Another way of looking at it is that our relative uncertainty about $\boldsymbol{C}'$ is greater than that about $\boldsymbol{C}$.

For computational purposes the rule can easily be manipulated to yield the equivalent condition for declaring a node to be a leaf in the absence of any other tests, i.e.,

$$p_{\max} \ge \frac{H_2(p_{\max})}{H(\boldsymbol{C})} - \frac{\log_2(n) - \log_2(n-1)}{\log_2(n-1)} \tag{19}$$

where $n$ is the number of nonzero class components at the node. In effect, one has a dynamic threshold on $p_{\max}$. The rule as outlined here is for the case where the class variable is uniformly distributed. The extension to the general case is given in Appendix IV. Note that this rule cannot be used unless $n \ge 3$, where $n$ is the number of classes.

No claims of optimality are made for this rule. Rather, we have a useful and practical test for establishing terminal nodes with the top-down tree design algorithm. In particular, the rule requires no externally supplied parameters, is independent of the noise level, and does not necessarily require large data sets to work properly. In the next section experimental results are given which indicate that the delta-entropy rule works well in practice.

To conclude this section, some final observations on the problem of termination tests are in order. Finding general termination rules for a top-down tree-design algorithm which can be applied to different problems with different noise levels is quite difficult. The fundamental problem lies in the concept of the greedy algorithm which by definition cannot "look ahead." The question is whether or not it is worth continuing to split. Clearly, a purely greedy algorithm can never be totally satisfactory in this regard, while on the other hand, all look-ahead strategies must be either suboptimal or *NP*-complete. To complicate the problem, varying levels of noise may be present which cannot be distinguished except by continuing to split the node further. In addition, the data set at the node may be too small to permit any confidence in the use of statistical tests. One might conjecture that the exact form of the termination rule to be used is inevitably dictated by the nature of the particular problem at hand, e.g., approaches such as "pruning" [12] (where subtrees are grown and later possibly removed by a "pruning" algorithm) may be appropriate in applications where a more complicated tree-derivation algorithm is acceptable. However, for the case of strictly top-down algorithms we have established from a theoretical point of view that threshold-type rules exhibit deficiencies and so confirmed earlier experimentally based conclusions. The delta-entropy rule is an alternative termination rule to overcome the deficiencies of the other currently known rules.

## VII. Simulation Results

Simulations were carried out by designing trees at various noise levels based on the following problem. Consider 16-segment alpha-numeric displays as shown in Fig. 8. The 16 segments correspond to 16 binary features. The 26 uppercase letters and 10 digits make up the 36 classes. This particular problem was chosen because, on the one hand, the parameters of the simulated data set (e.g., noise, number of samples) can be directly controlled while on the other, a relatively large number of both features and classes is exhibited. A more complicated problem with real data (as opposed to simulated data) might yield less insight into tree design *per se* than it would about the problem itself. The noise parameter $\epsilon$ has an easily interpretable meaning. Each segment has a probability $\epsilon$ of "doing the wrong thing" at the time the features are being measured, i.e., being off when it should be on or being on when it should be off. In fact, this is just a more complicated version of a seven-segment problem which has been simulated by others [11], [12]. The range of values chosen for $\epsilon$ during the simulations were 0.0 to 0.1 in increments of 0.01. It is interesting to note the reason for keeping $\epsilon \le 0.1$. With $\epsilon = 0.1$ we found that it was quite difficult for a human to recognize the characters, and the Bayes misclassification rate was estimated at about 0.25. It seemed reasonable to expect then that the area of greatest interest with respect to tree performance would be for $\epsilon \le 0.1$. The

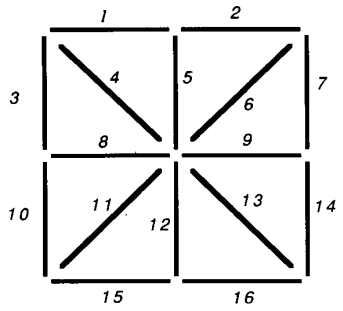


Fig. 8. 16-segment alpha-numeric display used for simulations with numbered segments corresponding to features.

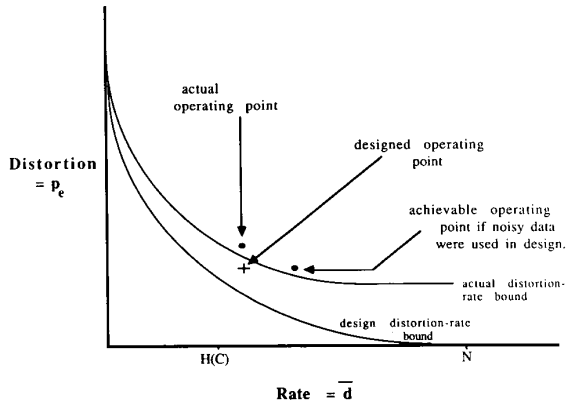


Fig. 9. Distortion-rate curve showing how good design data can lead to a bad classifier.

distribution of the alphabetic classes was taken to be that of English text as defined in [30], while the digits were simply assigned probabilities of 1/36 each. The number of data samples for each tree design was 5000. The misclassification rate $P_e$ was estimated by using the tree to classify an independent data set also of size 5000. Using a test set of size 5000 to estimate $P_e$ yields estimates of very high confidence levels for $P_e$, according to [31].

The noise level of the test set and the design set was the same. It has been experimentally observed by the authors and others [29] that using less noisy data than that encountered in practice while designing a tree classifier will actually lead to a higher misclassification rate than if the same type of data is used for both design and use. This conclusion can be inferred from the distortion-rate model as shown in Fig. 9. The situation is equivalent to having an artificially low distortion-rate bound during the design while actually operating with a bound which is higher up on the misclassification rate axis. Designing the tree with a given algorithm will result in $\bar{d}$ being fixed; then $P_e$ in practice will be higher than the desired $P_e$ or the resubstitution estimate. However, were one to design with data having the true (higher) value for $\epsilon$ one should be able to determine a lower $P_e$ for a correspondingly higher $\bar{d}$. This is what happens in practice, the artificial case yielding trees which are too short and do not exploit the redundancy in the features as the noise increases. On the other hand, if the situation is reversed and one is designing with noisier data than that encountered in practice (as might happen if at some later point in time one's feature measurements were to become less noisy) one can conjecture that for a designed $\bar{d}$ the actual misclassification rate in practice might well be lower than the resubstitution estimate but only because the $\bar{d}$ is significantly larger than it needs to be.

To provide an idea of what is involved, Fig. 10 shows a tree designed for the noiseless case, i.e., $\epsilon = 0$. The leaves (or terminal nodes) are labeled with the class decision at that leaf while the internal nodes are labeled with the feature (segment) to be evaluated at that node. Branches going to the right in the tree correspond to a segment being lit while those to the left mean that the segment is not lit.

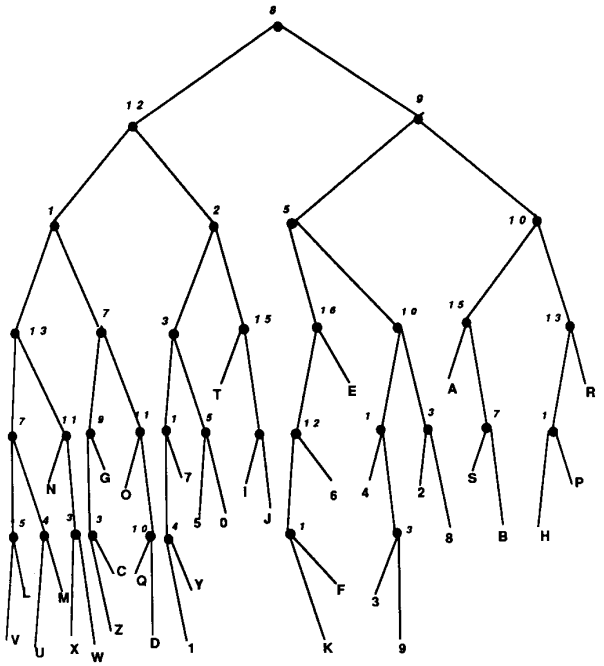

Fig. 10. Tree designed using delta-entropy rule for alpha-numeric problem.

It can easily be shown that for the special case of $\epsilon = 0$ the delta-entropy rule will always yield a "correct" tree in the sense that a 1–1 correspondence exists between leaves and classes. This property is necessary for any termination rule which is to be used on data where it is not known *a priori* whether or not $\epsilon = 0$. Threshold rules for example do not exhibit this property in general.

Fig. 11 shows the increase in average depth plotted against the noise level. Note that for $\epsilon = 0$ the average depth is only slightly greater than the entropy of the class distribution ($\bar{d} = 4.9692$, $H(C) = 4.7567$) and does not increase significantly above $H(C)$ as the noise level increases; i.e., even for $\epsilon = 0.1$, fewer than six of the 16

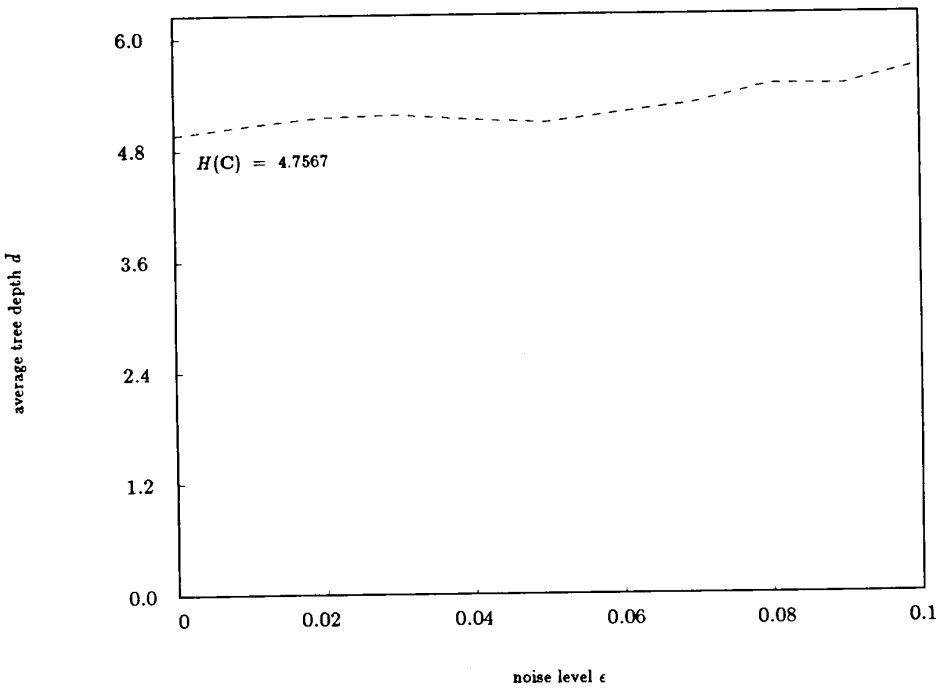


Fig. 11. Average tree depth, as designed for alpha-numeric problem versus noise.

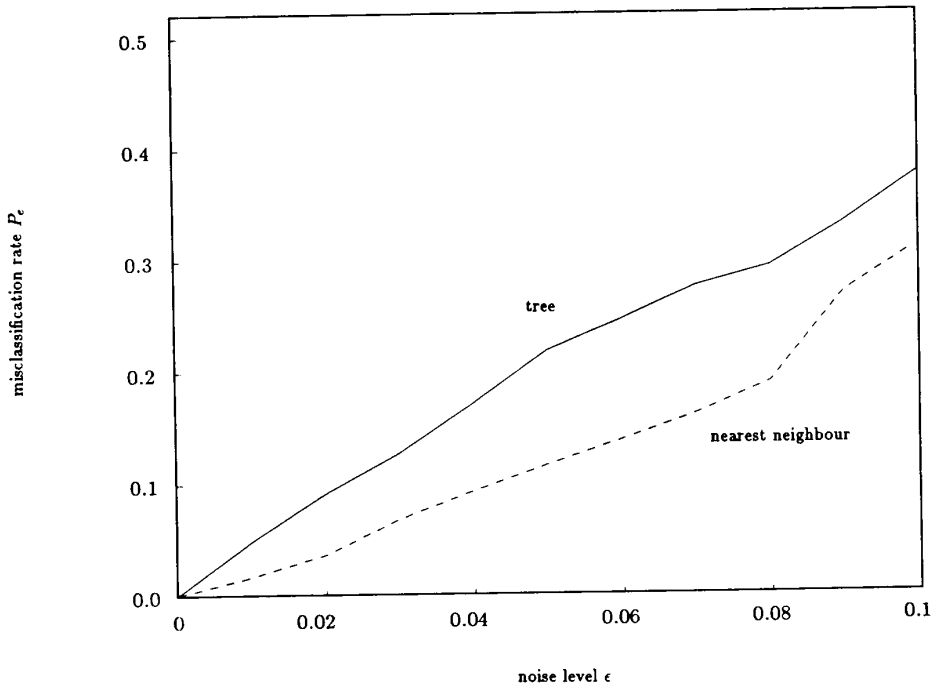


Fig. 12. Misclassification rates of tree classifiers and nearest neighbor classifiers for alpha-numeric problem, versus noise.

possible features need to be evaluated on the average. In fact, we found the delta-entropy rule to be quite conservative in our simulations, consistently yielding trees with an average depth quite close to the entropy of the class distribution. Average depth alone is not significant, however. Fig. 12 shows the estimated probability of misclassification for the trees of Fig. 11. On the same graph is a plot of the misclassification rate obtained using the same data with the nearest neighbor algorithm, i.e., where all 16 features were used. The tree classifier is seen to do very well in comparison with the nearest neighbor technique when in Fig. 13 we plot a merit function for each algo-

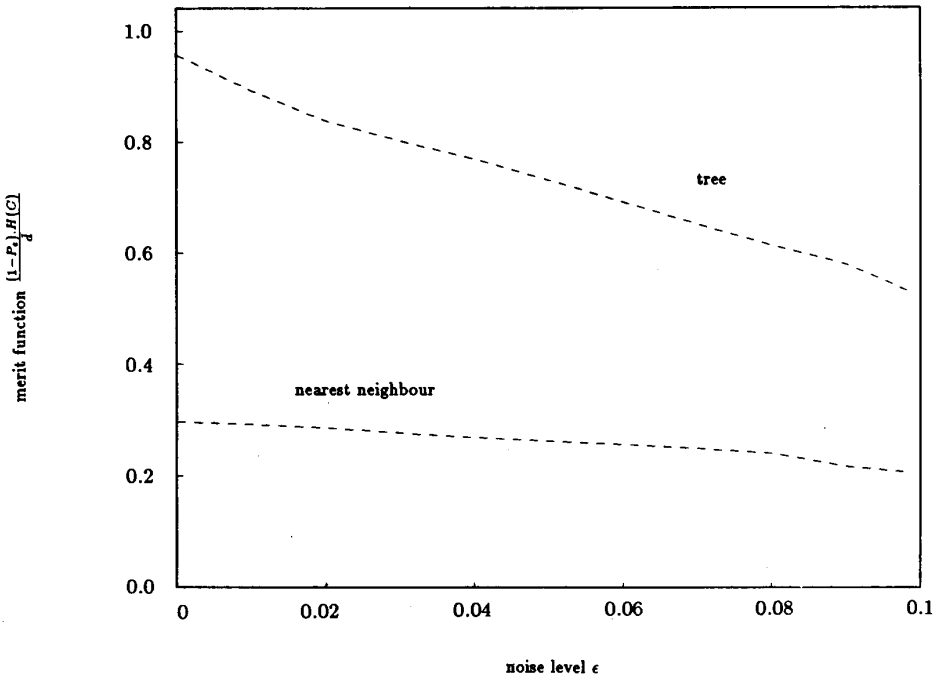


Fig. 13. Merit function for both tree and nearest neighbor classifiers for alpha-numeric problem versus noise.

rithm versus the noise level. The merit function was chosen to be

$$\frac{(1-P_e)H(\boldsymbol{C})}{\bar{d}}$$

where $\bar{d}$ is fixed at 16 for the nearest neighbor (even though effectively it is much greater than this). This merit function can be interpreted as a simple "benefit over cost" ratio where the benefit is $1-P_e$ and the cost is defined as $\bar{d}/H(\boldsymbol{C})$. Hence with $P_e=0$ and $\bar{d}=H(\boldsymbol{C})$, the merit function $=1$ which is its maximum attainable value. The tree classifier is clearly better for this criterion and indeed would be superior for any criterion which gives the average number of features evaluated at least equal weight with the misclassification rate.

That the tree classifier does well on this problem is a good indication of the fact that 16-segment alpha-numeric displays exhibit much redundancy. Similar success can be expected in classification problems where considerable redundancy exists in the feature data.

## VIII. Conclusion

Following a brief overview of designing tree classifiers, we established that the well-known top-down tree-design algorithm using mutual information is directly equivalent to a form of prefix coding. Using a communication theory analogy, we derived bounds on the maximum average information available at a node. This led, in turn, to a series of results involving average tree depth, probability of misclassification and the remaining average uncertainty. Using these results in conjunction with a recently proposed distortion-rate model [11] for tree classifiers, we looked at several experimentally observed phenomena which occur when designing tree classifiers and explained them in a theoretical context. In particular, we showed that threshold-type rules are not suitable for terminating the algorithm. We have proposed a new rule, the delta-entropy rule, which does not require externally supplied parameters and is robust in the presence of noise. Finally, a classification problem was simulated, and the results concur with the general theory outlined earlier.

The approach outlined in this paper is a basis for further work. In particular, we are investigating problems such as information-efficient quantization of real-valued attributes and designing trees when each attribute has an associated cost.

## Appendix I

*Proof of Theorems 1 and 2*

Let $\boldsymbol{Q}$ be a discrete random variable which can assume values from 1 to $m$ each with probability $q_k$, $1 \le k \le m$, $\Sigma_{k=1}^{m} q_k = 1$.

Consider a discrete memoryless channel as shown in Fig. 14. The input is $\boldsymbol{C}$, the class variable, which has an alphabet $\{c_1, c_2, \cdots, c_K\}$ as previously defined. The received output is $\boldsymbol{Q}$, the partition random variable, which has an "alphabet" of $m$ letters corresponding to the $m$-ary partition of $\boldsymbol{C}$. In the example in Fig. 14, $K=5$, $m=2$, and the partitioned subsets are $\{c_1, c_2\}$ and $\{c_3, c_4, c_5\}$. In other words, evaluating a feature at a node is equivalent to observing the output of a noisy channel where the possible outputs correspond to the different branches emanating from the node. One can thus write

$$I(\boldsymbol{C}; \text{node}_j) = I(\boldsymbol{Q}; \boldsymbol{C}) \tag{20}$$

$$= H(\boldsymbol{Q}) - H(\boldsymbol{Q}|\boldsymbol{C}). \tag{21}$$

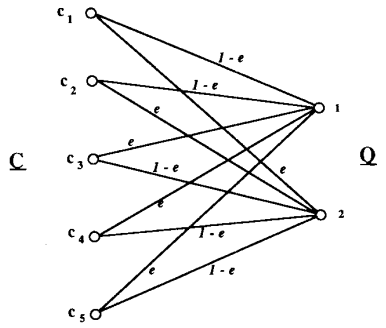


Fig. 14. Example of node modeled as discrete memoryless channel.

In particular, for no noise, $H(\boldsymbol{Q}|\boldsymbol{C}) = 0$, and this proves Theorem 1.

For the case with noise one can define an $m$-ary symmetric channel with transition probabilities such that

$$p(k|c_i) = \begin{cases} 1-\epsilon, & \text{if } c_i \in k \quad (22) \\ \dfrac{\epsilon}{m-1}, & \text{if } c_i \notin k \quad (23) \end{cases}$$

where $1 \le k \le m$ and $1 \le i \le K$. Then we have

$$\begin{aligned} H(\boldsymbol{Q}|\boldsymbol{C}) &= \sum_{k=1}^{m} \sum_{i=1}^{N_C} p(k,c_i) \log\left(\frac{1}{p(k|c_i)}\right) \\ &= \sum_{k=1}^{m} \sum_{i=1}^{N_C} p(k|c_i)\, p(c_i) \log\left(\frac{1}{p(k|c_i)}\right) \\ &= \sum_{k=1}^{m} \sum_{c_i \in k} p(c_i)(1-\epsilon) \log\left(\frac{1}{1-\epsilon}\right) \\ &\quad + \sum_{k=1}^{m} \sum_{c_i \notin k} p(c_i) \frac{\epsilon}{m-1} \log\left(\frac{m-1}{\epsilon}\right). \end{aligned} \quad (24)$$

Now since the partitions define disjoint subsets of the $c_i$

$$\sum_{c_i \notin k} p(c_i) = 1 - q_k, \quad (25)$$

and so

$$\sum_{k=1}^{m} \sum_{c_i \notin k} p(c_i) = \sum_{k=1}^{m} 1 - q_k = m-1. \quad (26)$$

This yields

$$H(\boldsymbol{Q}|\boldsymbol{C}) = (1-\epsilon) \log\left(\frac{1}{1-\epsilon}\right) + \epsilon \log\left(\frac{m-1}{\epsilon}\right). \quad (27)$$

## Appendix II
## Proofs of Theorem 3 and Theorem 3 Extended

*Proof of Theorem 3*

From Theorem 1, $I(\boldsymbol{C};\boldsymbol{Q}) = H_2(p)$, where $p$ is the probability of one of the two subsets (and $1-p$ is the probability of the other subset). Therefore, it suffices to prove that

$$H_2(p) \ge H_2\left(\max\left\{p_{\max}, \frac{1}{3}\right\}\right). \quad (28)$$

There are three cases to consider.

*Case 1)* $p_{\max} > 1/2$: Choose the partition probability $p = p_{\max}$, i.e. partition the maximum component of the distribution on its own. Therefore, in this case the inequality is actually an equality condition. This choice is, in fact, the optimal choice for the partition since the function $H_2(p)$ is convex about the point $p = 0.5$. However, this is incidental to the proof so it is sufficient to state that

$$H_2(p) = H_2(p_{\max}) = H_2\left(\max\left\{p_{\max}, \frac{1}{3}\right\}\right).$$

*Case 2)* $1/3 \le p_{\max} \le 1/2$: Now choose $p = p_{\max} + \sum_i p_i$ so that

$$\left|\frac{1}{2} - p\right| \le \left|\frac{1}{2} - p_{\max}\right| \quad (29)$$

if such $p_i$ exist, i.e., $p$ is closer to $1/2$ than $p_{\max}$. If such $p_i$ do not exist, then choose $p = p_{\max}$. In any case, because of the convexity of $H_2(p)$,

$$H_2(p) \ge H_2(p_{\max}) = H_2\left(\max\left\{p_{\max}, \frac{1}{3}\right\}\right).$$

*Case 3)* $p_{\max} \le 1/3$: There exist at least three other $p_i < p_{\max}$ since

$$\sum_{i=2}^{3} p_i + p_{\max} < \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1. \quad (30)$$

Consequently, it must be possible to construct an appropriate $p$ by adding $p_i$ to $p_{\max}$. More specifically, since all of the $p_i < 1/3$, some combination of $k$ $p_i$ must exist such that

$$\frac{1}{3} \le \sum_{j}^{k} p_j < \frac{2}{3}. \quad (31)$$

Let $p = \sum_j^k p_j$, so that

$$\frac{1}{3} \le p < \frac{2}{3}. \quad (32)$$

Now $\max\{p_{\max}, 1/3\} = 1/3$ by definition so that by the convexity argument one gets

$$H_2(p) \ge H_2\left(\frac{1}{3}\right) = H_2\left(\max\left\{p_{\max}, \frac{1}{3}\right\}\right).$$

From Cases 1–3 the statement is true in general:

$$\text{if } p_{\max} < \frac{2}{3}, \quad \text{then } I(\boldsymbol{C};\boldsymbol{Q}) > H_2\left(\frac{1}{3}\right) = 0.918 \text{ bits.} \quad (33)$$

*Proof of Theorem 3 Extended*

Theorem 2 established that

$$I(\boldsymbol{C};\boldsymbol{Q}) = H_2(p) - H_2(\epsilon).$$

Hence it suffices to prove that

$$H_2(p) \ge H_2\left(\max\left\{p_{\max} + \epsilon(1-2p_{\max}), \frac{1+\epsilon}{3}\right\}\right). \quad (34)$$

We have already seen in the noiseless case that

$$|p' - 0.5| \le \left|\max\left\{p_{\max}, \frac{1}{3}\right\} - 0.5\right| \quad (35)$$

i.e., we choose $p'$ so that it is at least as close to 0.5 as the greater of $p_{\max}$ or $1/3$.

Now by introducing noise, can the basic inequality be changed, i.e., can $p_{\max}$ or $1/3$ be closer to 0.5 than $p'$ when noise is accounted for? If not, then (35) remains true with the inclusion of noise and hence (34) holds.

It can easily be shown that

$$p+\epsilon(1-2p)=\max\left\{p_{\max}+\epsilon(1-2p_{\max}),\frac{1}{3}+\epsilon\frac{1}{3}\right\} \quad (36)$$

by the definition of $p$ ((9) Section IV). Hence it remains to show that

$$|0.5-(p'+\epsilon(1-2p'))|\le|0.5-(p+\epsilon(1-2p))| \quad (37)$$

given that

$$|0.5-p'|\le|0.5-p|. \quad (38)$$

Equation (37) can be manipulated to yield

$$\begin{aligned}&(0.5-(p+\epsilon(1-2p)))^2\\&\quad=(0.5-\epsilon)^2+2p(0.5-\epsilon)(2\epsilon-1)+p^2(2\epsilon-1)^2\\&\quad=(0.5-\epsilon)^2+(2\epsilon-1)\big((1-2\epsilon)p+p^2(2\epsilon-1)\big)\\&\quad=(0.5-\epsilon)^2+(2\epsilon-1)^2(p^2-p)\\&\quad\ge(0.5-\epsilon)^2+(2\epsilon-1)^2(p'^2-p')\\&\quad=(0.5-(p'+\epsilon(1-2p')))^2 \quad (39)\end{aligned}$$

## Appendix III

*Proof of Theorem 5*

We have

$$\begin{aligned}H(\boldsymbol{C})-H(\boldsymbol{C}|\boldsymbol{T})&=I(\boldsymbol{C};\boldsymbol{T})\\&=\sum_{j\in S}p(\text{node}_j)\,I(\boldsymbol{C};\text{node}_j). \quad (40)\end{aligned}$$

In addition, we have that

$$\bar{d}=\sum_{j\in S}p(\text{node}_j). \quad (41)$$

However, $I(\boldsymbol{C};\text{node}_j)\le 1-H_2(\epsilon)$ at any node by Theorem 2, and under the assumption made for part b) of this theorem one can use (9) in Section IV:

$$I(\boldsymbol{C};\text{node}_j)\ge H_2(p+\epsilon(1-2p))-H_2(\epsilon).$$

Hence by (40) and (41) one gets

$$\frac{I(\boldsymbol{C};\boldsymbol{T})}{1-H_2(\epsilon)}\le\bar{d}\le\frac{I(\boldsymbol{C};\boldsymbol{T})}{H_2(p+\epsilon(1-2p))-H_2(\epsilon)}$$

where $I(\boldsymbol{C};\boldsymbol{T})=H(\boldsymbol{C})-H(\boldsymbol{C}|\boldsymbol{T})$.

## Appendix IV

The form of the delta-entropy rule given in Section VI will only work when the original class distribution at the root node is uniform. In general, however, the distribution is nonuniform, and a more general form of the rule is required. The general form works with normalized probability components $p(c_i)$ relative to the root node, i.e., before applying the delta-entropy rule, divide each nonzero probability component $p(c_i)$ at the node by its original value $p_0(c_i)$ at the root node. Then renormalize the distribution. Effectively,

$$p(c_i)\quad\text{is replaced by}\quad\frac{p(c_i)}{p_0(c_i)}\,\frac{1}{\sum_{k=1}^{K}\frac{p(c_k)}{p_0(c_k)}}. \quad (42)$$

Then proceed to apply the simple delta-entropy rule as previously given to the normalized class distribution at the node. The reason for the normalization is to measure the *relative* decrease in uncertainty for each class.

## References


[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[2] G. F. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 55–63, Jan. 1968.

[3] A. Wald, *Sequential Analysis*. New York: Wiley, 1947.

[4] P. M. Lewis, "The characteristic selection problem in recognition systems," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 171–178, 1962.

[5] K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*. New York: Academic, 1968.

[6] L. N. Kanal, "Problem-solving models and search strategies for pattern recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 193–201, Apr. 1979.

[7] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Trans. Geosci. Electron.*, vol. GE-15, pp. 142–147, July 1977.

[8] S. Ganapathy and V. Rajaraman, "Information theory applied to the conversion of decision tables to computer programs," *Commun. ACM*, vol. 16, pp. 532–539, Sept. 1973.

[9] C. R. P. Hartmann, P. K. Varshney, K. G. Mehrotra, and C. L. Gerberich, "Application of information theory to the construction of efficient decision trees," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 565–577, July 1982.

[10] P. A. Chou, "A procedure to design efficient probabilistic decision trees," Master's thesis, Univ. of California, Berkeley, Dept. Electrical Engineering and Computer Science, 1983.

[11] P. A. Chou and R. M. Gray, "On decision trees for pattern recognition," presented at the 1986 IEEE Int. Symp. Information Theory, Ann Arbor, MI, Oct. 1986.

[12] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[13] S. Khuri, C. R. P. Hartmann, and P. K. Varshney, "Application of information theory to the conversion of probabilistic decision tables into efficient decision trees," preprint.

[14] D. Michie, S. Muggleton, C. Riese, and S. Zubrick, "RuleMaster—A second generation knowledge engineering facility," Radian Corporation, Austin, TX, Radian Tech. Rep. MI-R-623, Dec. 1984.

[15] R. G. Casey and G. Nagy, "Decision tree design using a probabilistic model," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 93–99, Jan. 1984.

[16] I. K. Sethi and G. P. R. Sarvarayudu, "Hierarchical classifier design using mutual information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, pp. 441–445, July 1982.

[17] Q. R. Wang and C. Y. Suen, "Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, pp. 406–417, July 1984.

[18] J. R. Quinlan, "Learning efficient classification procedures and their application to chess endgames," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Palo Alto, CA: Tioga, 1983, pp. 463–482.

[19] P. Argentiero, R. Chin, and P. Beaudet, "An automated approach to the design of decision tree classifiers," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, pp. 51–57, Jan. 1982.

[20] H. M. Dante and V. V. S. Sarma, "Automated speaker identification for a large population," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 255–263, June 1979.

[21] P. Smyth, "The application of information theory to problems in decision tree design and rule-based expert systems," Ph.D. dissertation, Dept. of Electrical Engineering, California Institute of Technology, Pasadena, CA, June 1988.

[22] R. M. Fano, *Transmission of Information*. Cambridge, MA: MIT Press, 1961, p. 187.

[23] R. G. Gallagher, *Information Theory and Reliable Communication*. New York: Wiley, 1968, p. 43.

[24] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, Sept. 1952.

[25] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision

trees is NP-complete," *Inform. Process. Lett.*, vol. 5, pp. 15–17, May 1976.

[26] J. K. Wolf and J. Ziv, "Transmission of noisy information to a noisy receiver with minimum distortion," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 406–411, July 1970.

[27] R. W. Conners and C. A. Harlow, "A theoretical comparison of texture algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 204–222, May 1980.

[28] V. A. Kovalevsky, *Image Pattern Recognition*, translated from Russian by A. Brown. New York: Springer-Verlag, 1980, p. 79.

[29] J. R. Quinlan, "The effect of noise on concept learning," in *Machine Learning*, vol. 2, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Los Altos, CA: Morgan Kaufmann, 1986, pp. 149–166.

[30] A. G. Konheim, *Cryptography, A Primer*. New York: Wiley, 1981, p. 16.

[31] W. H. Highleyman, "The design and analysis of pattern recognition experiments," *Bell Syst. Tech. J.*, vol. 41, pp. 723–744, Mar. 1962.