Machine Learning, 11, 7-22 (1993)

Coding Decision Trees

C.S. WALLACE CSW@CS.MONASH.EDU.AU
Computer Science, Monash University, Clayton 3168, Australia

J.D. PATRICK JON@CM.DEAKIN.OZ.AU
Computing & Mathematics, Deakin University, Geelong 3217, Australia

Editor: Dennis Kibler

Abstract. Quinlan and Rivest have suggested a decision-tree inference method using the Minimum Description Length idea. We show that there is an error in their derivation of message lengths, which fortunately has no effect on the final inference. We further suggest two improvements to their coding techniques, one removing an inefficiency in the description of non-binary trees, and one improving the coding of leaves. We argue that these improvements are superior to similarly motivated proposals in the original paper.

Empirical tests confirm the good results reported by Quinlan and Rivest, and show our coding proposals to lead to useful improvements in the performance of the method.

Keywords: decision trees, supervised learning, minimum message length, minimum description length, information theory

1. Introduction

Quinlan and Rivest (1989) have proposed a method for inferring Decision Trees based on the general inductive inference method known variously as Minimum Message Length (Wallace & Freeman, 1987) or Minimum Description Length (Rissanen, 1983). The present article is essentially a gloss on their paper, which we hereafter refer to simply as QR. Sections of QR will be referred to as QR2.1, equations as QR(3), etc.

While we support both the thrust and the major conclusions of QR, the paper contains what appear to us some errors of presentation that, while they turn out not to affect the body of QR, could lead to some confusion. Since QR presents an important advance in the theory of decision tree inference, it seems desirable to try to clarify the points concerned. We also suggest technical modifications of the QR algorithm that appear to have some advantages.

The Decision Tree problem is an instance of the general regression problem, which may be described as follows. The given data refer to a set of N things, usually representing a random sample from a large population. There is a defined set of "attributes" or "independent variables." For each thing, the data give a vector comprising a value for each variable. There is a further, distinguished variable, the "dependent variable," and the data give its value for each thing. There is also a family of functions defined on the set of possible independent variable vectors. The general regression problem is to find, within the family, that function of the independent variables that best predicts the value of the dependent variable. It is hoped that if a function is found that is a good predictor of the dependent variable over the sample set, it will give useful predictions for new things from the same population.

The general regression problem takes a number of forms depending on the nature of the variables involved, the family of possible predictor functions, and the criterion used to judge predictive success. For instance, in the classical linear regression problem, all variables are real-valued, the family is the family of linear combinations of the independent variables, and the usual criterion is the mean squared difference between the value of the predictor function and the dependent variable.

In the Decision Tree problem of Quinlan (1986), the dependent variable is categorical, and its value is termed the "class" of a thing. The predictor function is a rooted tree that sorts things into "categories" by routing each thing from the root node along branches of the tree until it reaches a leaf node. Each non-leaf node is labelled with an attribute. If the attribute is discrete, one branch leaves the node for each possible discrete value, and a thing reaching the node takes the branch given by its value for the attribute. Conventionally, we label the possible values with the integers 1, 2, 3, etc., and draw the branches with 1 on the left. If the attribute labelling a node is real, the node is further labelled with a "cut value." A thing reaching the node takes the left branch if its value for the attribute is less than the cut value; otherwise it takes the right branch.

Each leaf node defines a category. Note that the category to which a thing is assigned is a function of its attribute values, i.e, the independent variables, and not of its class.

A tree that sorts the sample things into categories correlating closely with their classes can be used for prediction. Any new thing of unknown class can be placed in a category, and a prediction of its class can be made by inspecting the classes of the sample things in that category.

2. The induction principle

Any theory or hypothesis T about a body of data D can be used to encode the data as a binary string. Following classical coding theory (Shannon & Weaver, 1949; Rissanen & Langdon, 1981), the length of the code string using an optimally efficient code is given by

$$-\log P(D \mid T) \text{ (logs to base 2)}$$

where $P(D \mid T)$ is the probability of obtaining the data D if T is true. The resulting string is decodable only to a receiver who knows the theory T. If the theory is novel, the string must therefore be preceded by a string that specifies T using some code agreed a priori. The length of the string describing T in an efficient code may be identified with $-\log P(T)$, where $P(T)$ is the a priori probability of the theory. Conversely, adoption of any non-redundant code for specifying one of a set of possible theories is equivalent to adoption of a prior probability distribution over the set. The length of the total message is then

$$-\log[P(T)P(D \mid T)] \tag{1}$$

The minimum message length principle infers from data D the theory that minimizes the total message length. Since, by Bayes's Theorem,

$$P(T)P(D\,|\,T) = P(D)P(T\,|\,D) \tag{2}$$

where $P(D)$ is the (constant) probability of getting D and $P(T\,|\,D)$ is the posterior probability of the theory T given the data D, the induction principle is, at least for countable sets of possible theories, formally identical to the Bayes rule that selects the theory of highest posterior probability.

Unfortunately, QR confuses this simple correspondence. In QR2.1, it is argued that if a string of length t encodes some statement (e.g., a theory), the probability associated with the statement is

$$\left(1 - \frac{1}{r}\right)\left(\frac{1}{2r}\right)^t \tag{QR(1))}$$

where $r > 1$ is some constant, which QR chooses as 2. This result is obtained by constructing a normalized probability distribution over the set of finite binary strings in which probability decreases geometrically with string length. Choosing $r = 2$ gives to a string of length t a probability

$$2^{-(2t+1)} \tag{3}$$

This assignment of probabilities is inapplicable to the strings of concern in MML inference.

If a code is devised to allow the binary encoding of any one of a set of statements, the set of strings corresponding to the set of statements is not an unrestricted set of finite binary strings. It must have the prefix property: no string of the set may be a prefix of any other. If this constraint is not observed, a received string cannot be unambiguously decoded (Hamming, 1980).

For any set of strings having the prefix property, where string i has length t_i, it is well known that

$$\sum_i 2^{-t_i} \leq 1$$

and for any non-redundant code, equality obtains. Thus it is possible and natural to associate with a *code* string of length t the probability 2^{-t}, as we have done in equating the length of a message naming theory T with $-\log P(T)$, and so on. In fact this association necessarily follows for the strings of a code of minimal expected length.

Luckily the impact on this error in QR2.1 is minimal. In most of the paper, the same value $r = 2$ is used in considering the coding of both theory and data, and all the codes considered have the prefix property. Thus when QR considers minimization of total message length, the quantity considered is just twice the correct value, so the correct minimizing tree is found. In QR8, experiments are reported in which different values for r are used in the coding of theory and data, but QR warns that there is no firm justification for this choice.

The known theoretical results (e.g., Wallace & Freeman, 1987; Barron & Cover, 1991) suggesting the optimality of MML inference give no basis for treating the bits used to encode

the theory differently from the bits used to encode the data given the theory. The small advantage found in QR table III for a higher weight for data bits may be an artifact of their use of a somewhat inefficient code for the theory, as discussed below, having the effect of an overweighting of the theory description. QR suggests reasons for using unequal weights in some circumstances, but we agree that the question remains open.

3. The decision tree message

The message of concern in a regression problem is a message that conveys the sample things' values of the dependent variable to a receiver who already knows their attribute values. The "theory" part of the message specifies a predictor function, normally interpreted as predicting for any attribute vector a probability distribution over dependent variable values; the "data" part encodes the things' dependent variable values, using for each thing a code optimized for the probability distribution predicted from its attributes.

For decision trees, there are three elements to consider in the message. First is the coding of the structure of the tree, including the labelling of each non-leaf node with an attribute and (for real attributes) cut value. Second, there is the encoding of the prediction associated with each category (leaf node). Third, there is the encoding of the things' classes, using for each thing a code based on the prediction associated with the thing's category. The order in which elements of these three kinds appear in the message is not necessarily first kind, second, and then third kind. The coding used in QR intersperses part of the prediction for each category within the description of the tree structure. Nor is it necessarily the case that an efficient coding will clearly distinguish between elements of the second and third kinds. In what follows, we describe the QR message in a slightly rearranged form, designed more clearly to separate the three kinds of element. The rearrangement does not affect the length or intelligibility of the QR message in any way, and is made purely for clarity of exposition.

3.1. The structure message

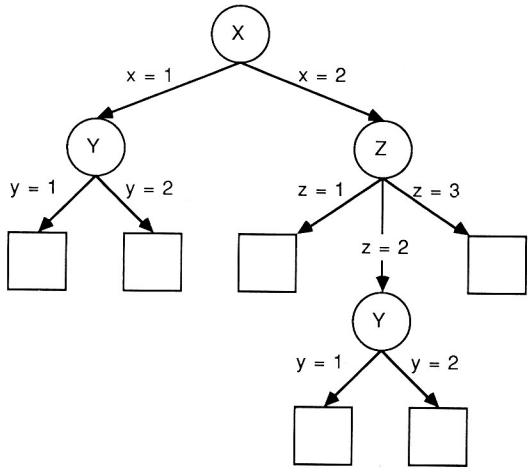
QR4.4 first describes what we will term the "simple" code for the tree structure. Initially assume all attributes are discrete, so there are no cut values involved.

The code for a leaf is binary 0.

The code for a non-leaf node is binary 1, followed by a code string for the attribute labelling the node, followed by the codes for the daughter nodes for each possible attribute value in increasing order.

The code for the decision tree is the code for its root.

For example, if the attributes X, Y, and Z have, respectively, 2, 2, and 3 values, the tree



is encoded as

1 *X* 1 *Y* 0 0 1 *Z* 0 1 *Y* 0 0 0

As QR points out, if there are n attributes, the length of the code for the attribute labelling the root is $\log n$, but the codes for attributes labelling deeper nodes will be shorter. At any node, only those discrete attributes that have not appeared in the path from the root to the node are eligible to label the node. Thus, the length of "X" in the above message is log 3 bits, the length of the first "Y" is log 2 = 1 bit, and the length of the second "Y" is zero.

Note that we have removed for later discussion the "default class labels" that appear in the QR description of the tree coding, since these are elements of the second kind.

If some attributes are real, the simple code must be followed by a list of cut values, and in coding attributes labels, one must note that a real attribute does not become ineligible once used.

QR correctly asserts that the simple code is inefficient when attributes have more than two values. Being non-redundant, the code is efficient for some prior probability distribution over the set of possible trees, but the implied distribution is unrealistic except for uniformly binary trees. Suppose all attributes have $b > 2$ values, so the trees are uniformly b-ary. Ignore for the moment the limitations on tree size imposed by the finite numbers of attributes and sample things, and consider the purely structural string, omitting the attribute labelling and cut values. It comprises just binary digits, a 0 for a leaf and a 1 for a non-leaf, the latter being followed by the b code strings for the b daughters. Suppose we expect 1s to occur with some fixed probability p independent of preceding digits. Coding techniques exist in effect allowing each 1 to be coded with $(-\log p)$ bits, and each 0 with $(-\log(1 - p))$ bits, and such a code is optimal if indeed the probability of 1s is p. Let $v(p)$ be the expected size, in nodes, of the subtree rooted at some node. If the node is a leaf (0), the size is 1. If it is non-leaf (1) the expected size is $(1 + bv(p))$, since each of the b daughters is the root of a subtree of expected size $v(p)$. Hence

$$v(p) = (1 - p) + p(1 + bv(p))$$

$$v(p) = \begin{cases} 1/(1 - bp) & (bp < 1) \\ \infty & (bp \geq 1) \end{cases}$$

Further, if $w(p)$ is the probability that the subtree rooted at some node is finite, then $w(p)$ is the smallest positive root of the equation:

$$w(p) = (1 - p) + p(w(p))^b$$

The smallest positive root is one iff $bp \geq 1$.

For instance, if $p = 1/2$, $b = 3$, we find $w = 0.62$. Hence the simple code, which encodes leaves and non-leaves each with one bit, in effect assuming $p = 1/2$, is inefficient for ternary trees, since it implies a distribution giving probability 0.38... to the set of infinite trees. More importantly, in all the strings encoding finite trees, the fractions of 1s is about 1/3. Hence use of the simple code, implicitly assuming $p = 1/2$, incurs an average additional length of

$$1 + (1/3)\log(1/3) + (2/3)\log(2/3) = 0.089...$$

bits per node.

For binary trees, the simple code is satisfactory. The set of infinite trees has measure zero ($w(0.5) = 1$) and the expected tree size is infinite. Of course, the finite numbers of attributes and things means the tree cannot exceed a certain size, so the code is slightly redundant in that it does not consider these limitations. However, the probability assigned by the implied distribution to trees larger than the limits imposed by the data is typically negligible.

QR proposes instead to encode the string of leaf/non-leaf choices as follows, using the fact that the string must have more zeros than ones. First, the number n of nodes is stated (QR omits to specify this coding, but it seems to be assumed). Then the number k of interior nodes is stated using $\log[(n + 1)/2 + 1]$ bits, since $k < n/2$; then the position of the k ones in the n digits of the structure string is stated using

$$\log \begin{pmatrix} n \\ k \end{pmatrix}$$

bits. (We think $(n + 1)/2$ should be replaced by $(n - 1)/2$.) This is not an efficient code, since many of the resulting code strings cannot represent a tree. For instance, consider binary trees of 5 interior and 6 leaf nodes. The QR proposal, having stated 11 and 5, then specifies one of

$$\begin{pmatrix} 11 \\ 5 \end{pmatrix} = 462$$

possible structure strings. However, from QR(14), there are only

$$\binom{10}{5}/6 = 42$$

binary trees with 5 interior nodes.

The inefficiency arises because, in a structure string representing a binary tree, the running count of zeros may not exceed the running count of ones until the end of the string. Thus,

1 1 0 0 1 0 1 1 0 0 0 represents a tree, but
1 1 0 0 1 0 1 0 0 1 0 does not.

The same kind of inefficiency occurs for higher arity.

Instead, we propose a modified form of the simple code. We retain its general form, but instead of using exactly one bit to specify whether a node is leaf or not, use for each node other than the root, p = the reciprocal of the arity of the parent of the node. Thus, in a uniform b-ary tree, each node other than the root is regarded as having probability $1/b$ of not being a leaf. The presumption that a node resulting from a many-way split of the things is more likely to be class-homogeneous (and hence a leaf) than one resulting from a binary split seems not unreasonable. The code is identical with the simple code for uniform binary trees, but for any arity retains the properties of implying zero measure for the set of infinite trees, and infinite expected tree size. Unlike the QR proposal, the code is not redundant.

The probability that the root is a leaf remains to be determined. Since in practice we hope to get a useful tree, this probability should be low. We have taken it as the reciprocal of the number of attributes, on the grounds that the more attributes there are, the better is the chance of finding one that is useful.

QR encodes the cut points for real attributes by naming one of the n known values at a node using a code of length $\log n$. When n is large, it seems unlikely that the predictive power of the cut will be much affected by changing the cut by one or two values, so we expect the QR code will typically waste a few bits in specifying the cut with needless precision. Indeed, the discussion in QR4.6 makes the same point. We adopted a form of an alternative discussed in QR4.6. A cut at the median is encoded with one bit, a cut at either the first or third quartile with three bits, a cut at any odd octile with five bits, and so on. The implied probabilities and code lengths are slightly adjusted for normalization when n is not a power of two.

3.2. The category message

The second and third elements of the complete message are expected, respectively, to announce a "theory" about the distribution of classes within each category, and to encode the things' classes using a code optimal for the theory. We may consider each category separately. We first discuss problems with just two classes, and consider a category which, for the sample set of things, contains n things of which k are class A and $(n - k)$ are class B.

Since the things are regarded as independently sampled from some population, any "theory" about the class distribution in the category can be represented by the probability $p = P(B)$ that a thing will be found to have class B. We might therefore expect the coding for the sample data in a category to have the form:

(i) Statement of an estimate $\hat{p}$ of p.
(ii) A string of A and B symbols, where A is encoded with length $-\log(1 - \hat{p})$, B with length $-\log \hat{p}$. (Note that the receiver knows n.)

As shown by Wallace and Boulton (1968), a coding of this form can be constructed and minimized. The length depends in part on the prior probability density assumed for p. For a prior density uniform in (0, 1), the length is approximately

$$\frac{1}{2}(\log(n/12) + 1) - \left(k + \frac{1}{2}\right)\log \hat{p} - \left(n - k + \frac{1}{2}\right)\log(1 - \hat{p}) \tag{4}$$

which is minimized by the estimate

$$\hat{p} = \left(k + \frac{1}{2}\right)/(n + 1) \tag{5}$$

Expression (4) is an approximation, and behaves poorly for very small n.

An alternative coding, which we will term the incremental code, does not attempt to state an estimate of p. Instead, each A or B symbol is encoded in turn with length $-\log(1 - q)$ for A, $-\log q$ for B, where q varies from symbol to symbol.

To encode the $(j + 1)th$ symbol, after i $B's$ and $(j - i)A's$ have been encoded, we use

$$q(i, j) = \int_0^1 p\, P(p \mid i, j)\, dp \tag{6}$$

where $P(p \mid i, j)$ is the posterior density of p given i and j,

$$P(p \mid i, j) = \binom{j}{i} p^i(1 - p)^{j-i}\, P(p \mid 0, 0) \tag{7}$$

and $P(p \mid 0, 0) = P(p)$ is the prior density of p.

This choice of q minimizes the expected length.

For $P(p)$ the Beta prior

$$P(p) = p^{\alpha-1}(1 - p)^{\beta-1} / B(\alpha, \beta) \tag{8}$$

it is easily shown that

$$q(i, j) = (i + \alpha)/(j + \alpha + \beta)$$

We will be interested in symmetric Beta priors for which $\alpha = \beta$, giving

$$q(i, j) = (i + \alpha)/(j + 2\alpha) \tag{9}$$

In particular, the uniform prior given by $\alpha = 1$ gives

$$q(i, j) = (i + 1)/(j + 2) \tag{10}$$

For the uniform prior, the total length of the code for a string of n symbols including k *B's* is

$$\log(n + 1) + \log \binom{n}{k} \tag{11}$$

which may be interpreted as the cost of specifying k in the range 0 to n, plus the cost of specifying which selection of k out of n symbols are *B's*. This code is described in QR4.2. However, the code actually proposed in QR4.5 is slightly different.

Instead of coding k, the number of *B's*, on the assumption that k is equally likely a priori to take any value from 0 to n, QR uses one bit to specify which of the two classes *A* or *B* is the more numerous ("default") in the category, and then encodes the number of the less numerous (either k or $(n - k)$) using $\log(n/2 + 1)$ bits, where $n/2$ appears to be rounded down. For n odd, the resulting length is exactly the same as for the incremental code, i.e., (11), but for n even, the length is slightly greater, because for even n,

$$1 + \log(n/2 + 1) > \log(n + 1).$$

The increase is related to the fact that for even n, $k = n/2$, the QR scheme provides two possible codings: either class could be named the default.

The choice of this scheme over the incremental code may be motivated by the realization that use of the incremental code results in an incomplete "theory." The incremental code does not specify an estimate p of the class distribution. It merely encodes the observed sample classes efficiently. In the terms of section 2, a message for a category advancing a theory T about p, and using it to encode the class data D, should from (2) have length

$$-\log P(D) - \log P(T \mid D)$$

The incremental code has length just $-\log P(D)$. Thus it is shorter than a message involving inference, such as (4). The QR scheme has the appearance of making an inference, in that it uses one bit to specify the more numerous class, and QR describes this bit as being part of the tree structure message. However, the appearance is deceptive. Were the announcement of the "default class" truly an inference about the population from which the sample things were drawn, e.g., the inference that $p > \frac{1}{2}$ for default class *B*, then the coding of things' classes should be based on a code optimized for that inference. Such a code would still permit a coding, albeit a longish one, for a set of classes in which *B* is in the minority. The assumption that $p > \frac{1}{2}$ in the population does not imply that $k > n/2$ in every sample.

A true "inference" code, whose length is (4), can still encode samples where k/n is very different from the estimate $\hat{p}$, but the QR scheme cannot encode a sample with $k < n/2$ if it has stated that B is the default. The QR scheme is a disguised form of the incremental code, and seems to have no advantage over it.

Having pointed out that neither the QR nor the incremental code actually involves inference of the class distribution in a category, we now concede that in practice its use is entirely sensible in this context. It has been shown (Boulton & Wallace, 1969) that the expected difference between the lengths of the inference and incremental codes is very small: about 0.24 bits for two classes. The main interest is in the inference of the tree structure rather than the within-category class distribution, so the fact that the incremental code length is easy to compute, and is well behaved even for small n, makes it a sensible choice. For just these reasons, we have previously used it in a rather similar context in the inference of finite-state grammars (Georgeff & Wallace, 1984).

3.3. The category prior

The incremental code generalizes gracefully to more than two classes. If there are M classes, and in the first j things of a category, i_m have had class m, the class of the $(j + 1)th$ thing is encoded assigning a probability

$$q_m = (i_m + \alpha)/(j + M\alpha) \tag{12}$$

to the class m, where we have assumed a generalized symmetric Beta prior over the unknown class probabilities $[p_m]$ proportional to

$$\prod_m p_m^{(\alpha-1)} \left(\sum_m p_m = 1, p_m > 0 \right) \tag{13}$$

QR 4.5 proposes a different generalization based on a recursive extension of the notions of "default" and "exception." The proposal is not elaborated, but seems to imply a probability distribution over the class numbers k_m for a three-class problem inversely proportional to

$$(2n/3 + 1)((n - k_{\max})/2 + 1)$$

where $k_{\max} = \max_m(k_m)$.

We do not consider this an attractive proposal, since it does not follow from any regular prior density over the class probabilities.

Returning to the Beta prior (13), we now consider whether the uniform prior given by $\alpha = 1$ is appropriate. The aim of constructing a decision tree is, loosely, to produce leaf categories each predominantly of a single class, i.e., of high class purity. We therefore

have a strong prior expectation that most of the categories given by any interesting tree will be fairly pure. If they were not, we would not get a short encoding of the data. The uniform prior does not express such an expectation. However, values of α less than one give Beta priors placing greater weight on fairly pure distributions. In the limit as α approaches zero, the prior becomes singular, placing all weight on the M possible single-class distributions. We propose a symmetric Beta prior with $0 < \alpha < 1$ as an appropriate basis for coding the classes of things in a category. Note that incremental calculation of the message length via (9) is straightforward.

It is difficult to propose any fixed value of α, at least until wider experience with real-world data is obtained. However, the data themselves can be used to estimate α. Using the method of QR6, a tree can be constructed assuming, say, $\alpha = 1$. The QR method grows the tree from the root, at each node selecting a test attribute on the basis of minimizing the information needed to select the attribute, define its cut value if real, and encode the classes of things in the resulting children nodes. This process is continued until either every leaf node is pure, or there is no attribute eligible to split. Having grown this "full" tree, QR then prunes back splits that proved not to improve the total message length.

We suggest instead that, having grown the full tree with $\alpha = 1$, say, the best pruned form and message length can be computed for various values of α, and the best value of α found. This is a relatively quick process, as the full tree is not altered. It gives a maximum likelihood estimate of α for the pruned tree.

Having estimated α, the tree-building process can be repeated using the estimated α, and may result in a different full tree. If so, α is re-estimated and so on until no improvement is obtained. Convergence is usually found in two or three iterations.

4. Variations

4.1. Lookahead

The basic QR tree-building heuristic selects as the test attribute at a node the attribute that would minimize the message length were the resulting child nodes leaves. Following a suggestion in QR6, we implemented a user-selectable lookahead, in which attribute selection considers the possible splitting of the child nodes. Taking the basic QR heuristic as zero-level lookahead, we have used up to four-level lookahead on some data sets.

To keep some rein on computing time, when a real attribute is considered the cut value is selected using no lookahead.

4.2. A dangerous optimization

At the end of QR4.4, it is suggested that if a binary split results in two leaves, some code length may be saved by using the constraint that the default classes in the two leaves must be different, since otherwise the split would be useless. This is not so. Suppose there are two classes A and B, and that a node N receives 75 A's and 24 B's. If a binary attribute is found that would cause a split into child nodes X and Y with class profiles (50A, 0B)

and (25*A*, 24*B*), the split may well be worthwhile even though both leaves have default class *A*. Splitting node *N* does not change any class prediction for new data: the prediction will be *A* for any case reaching node *N*. However, the objective sought by the MML algorithm in both QR and our versions is not the correct classification of the maximum number of cases, but rather the minimization of the amount of information needed to determine the class once the category is known. The split of *N* allows half the cases (those going to X) to have their class unambiguously defined by the category. The remainder, going to Y, are left with less hint of class than at node *N*, but overall there is a net reduction in uncertainty. Such splits are made in our algorithm, and should not be excluded in the QR version. For similar reasons, we believe that a simple count of errors made on test data by predicting the "default" class in each category is a poor measure of the predictive power of a tree.

4.3. An alternative category prior

QR4.2 suggests that the number k of exceptions in a (two-class) category might be better encoded using a code that favors small values of k. Our Beta-prior proposal has this effect for small α. QR suggests a code, described in QR appendix *A*, which encodes an integer k with a code length slightly greater than $\log k$, implying a prior distribution over the integers approximately proportional to $1/k$, but normalized. We do not support this suggestion, as it implies a prior probability of getting k exceptions in a category that is almost independent of the number of things in the category.

5. Test results

Results are given for six data sets. Five are ones used in QR (our "XD6" is QR "Prob-Disj"), and the Geographic set relates to the agricultural utility of land.

Discordant has two classes and 3772 things, but only 58 members of one class. Its attributes are mostly binary or real, with values missing for some attributes. Geographic has four classes and 106 things, and attribute arity up to eight. XD6 has two classes, 600 things, and ten binary attributes.

Tables 1, 2, and 3 give results for three sets, using various combinations of structure code, category prior, and lookahead. "Simple" is the simple structure code using one bit to indicate leaf or non-leaf; "Modified" is the structure code proposed in section 3.1, expecting a split with probability 1/(parent arity). Entries for $\alpha = 1$ show use of the QR incremental category code; entries for $\alpha < 1$ show use of a Beta prior with α optimized. The columns headed "Interior," "Leaves," and "Bits" give the numbers of interior and leaf nodes and the total message lengths for trees encoding the complete data sets.

The columns headed "Error %" and "Prediction cost" refer to trees built using two thirds of each data set, and give the error rates observed when the resulting trees were used to predict the classifications of the remaining third of the data. "Error %" gives the percentage of test things whose classes differed from the majority class of their category. "Prediction cost" is a more useful measure, giving the length in bits of a message encoding the classes of the test things, where the class of each thing is encoding using a code based

Table 1. Tests on "Discordant."

Code	Lookahead	α	Interior	Leaves	Bits	Error %	Prediction Cost
Simple	0	1	5	8	292	1.29	93
		0.28	7	10	284	1.29	94
	2	1	7	10	285	1.37	87
		0.28	8	11	275	0.89	77
Modified	0	1	3	6	291	—	—
		0.31	3	6	284	1.44	89
	2	1	6	9	284	—	—
		0.30	8	11	275	0.89	78
No	Tree	1	0	1	441	1.37	130

Table 2. Tests on "Geographic."

Code	Lookahead	α	Interior	Leaves	Bits	Error %	Prediction Cost
Simple	0	1	2	10	155	39	32
		0.035	7	15	118	26	20
	2	1	3	15	147	39	32
		0.022	5	17	107	26	19
Modified	0	1	1	9	149	42	36
		0.039	6	14	125	32	24
	2	1	3	15	136	26	28
		0.039	3	15	96	26	19
No	Tree	1	0	1	206	65	58

Table 3. Tests on "XD6."

Code	Lookahead	α	Interior	Leaves	Bits	Error %	Prediction Cost
Simple	0	1	17	18	440	25	167
		0.48	17	18	436	25	170
	1	1	14	15	442	19	135
		0.761	14	15	442	19	135
	2	1	17	18	425	—	—
		0.60	17	18	423	—	—
	4	1	14	15	420	15	120
		0.43	15	16	415	15	120
No	Tree	1	0	1	568	37	188

on the class distribution of the training set in its category. The measure is minus the log probability of obtaining the test classes under the hypothesis represented by the training set tree.

Note that these predictive tests use trees built with a randomly selected subset of the data, so the trees in general have slightly different structure and optimum α from those built using all the data. Each table also gives a total message length and prediction measures for the "null tree" comprising only the root.

5.1. Discussion

For Discordant and Geographic, the results clearly favor the use of a Beta prior with $\alpha < 1$. This prior gave shorter total message length, fewer prediction errors, and lower prediction cost. XD6 showed little or no difference between priors, and the optimal value of α of 0.5 or more represents little expectation of category purity. In fact, the artificial data of XD6 are generated so that a division into categories according to the Boolean function of attributes 1 to 9, namely,

a1.a2.a3 + a4.a5.a6 + a7.a8.a9

should give categories with 90% purity, but the above function was not discovered by the algorithm from the limited data available. A tree fully representing this function would need about 80 nodes.

The "Simple" and "Modified" structure codes differ only for nodes giving a split of more than two branches, and therefore are identical for XD6. The Discordant attributes are mostly real or binary, giving more than two child nodes only when a third node was introduced to accommodate "missing" values, so it is not surprising that the codes differed little on these data.

The Geographic data has some multi-valued attributes, and the modified code gave the shortest total message length for these data. Note, however, that the potential benefit of the modified code was not realized without lookahead.

The reductions in message lengths resulting from the use of a tree, or from a coding modification, may seem rather unimpressive. For instance, the use of the modified code on Geographic resulted at most in a reduction of total message length from 107 to 96 bits. However, the message length is minus the log of a probability, so this reduction means the posterior odds ratio in favor of the hypothesis using the modified code is about 2000. Similarly, the reduction of "prediction cost" from 87 to 77 resulting from the adjustment of α on Discordant (lookahead 2) means the test data favored the adjusted tree by an odds ratio of 1000.

The total improvement on Geographic on the original QR algorithm produced by the new structure and category codes and lookahead is 59 bits, corresponding to an odds ratio of order 10^{17}.

Table 4 summarizes results on five data sets tested in QR. The QR results are headed MDLP, and ours are headed MML. For brevity, only the modified code, lookahead2 MML

Table 4. Comparison with QR.

	MDLP		MML		
Data Set	Size	Error %	Size	Error %	α
Hypo	11	0.6	11	0.6	0.03
Discordant	15	1.9	11	0.9	0.30
LED	83	26.9	37	26.1	0.19
Endgame	15	17.9	10	15.2	0.23
XD6	17	20.5	16	15	0.43

results are given. "Size" is number of leaves. The error rates, both for QR and our results, are based on a random selection of one third of the cases as test data. Our random selections presumably differ from QR's.

The new results are generally superior. Part of the improvement is due to the lookahead used, but even without lookahead, the revised code usually gave slightly better results. The differences are not dramatic: our algorithm is only a slightly tidied-up version of QR, and our tree code would be shorter than QR's only by one bit or less per node.

6. Conclusions

Quinlan and Rivest (1989) have shown that minimum message length inference can be applied to the decision tree problem, and is at least competitive with other methods. There seems to be a flaw in their presentation of the relation between code length and probability, but this has no real impact on their work.

The full power of MML inference is obtained only when careful attention is paid to the optimization of coding techniques. QR correctly identifies an inefficiency in the "simple" code for non-binary trees, but proposes an alternative that itself is inefficient. We have instead shown a modification of the simple code restoring its efficiency.

Adoption of a particular code for encoding the "theory" (in this case a tree) induced from the data is equivalent to accepting a certain prior probability distribution over the set of possible theories. The code should be chosen not only to be technically efficient but also to imply a prior distribution that is reasonable in the problem domain.

The QR proposal, while appearing to infer the class distribution in a category, in fact does not. We argue that, in this context, the omission is acceptable, and show that by abandoning the fiction of an inference, a more efficient and flexible coding for classes can be used. The incremental code based on a symmetric Beta prior generalizes to multiple classes more gracefully than the QR scheme, and allows a more realistic model of the expectation that categories will have high purity.

A Fortran 77 program incorporating these amendments and implementing limited lookahead has been written, and trials on several data sets support the value of the amendments. The program is freely available for research from C.S. Wallace.

Acknowledgment

This work was supported by Australian Research Council grant A49030439. Some of the data sets were made available by W. Buntine and J.R. Quinlan.

References

Barron, A.R., & Cover, T.M. (1991). Minimum complexity density estimation. *IEEE Transactions on Information Theory, 37*(4), 1034–1054.

Georgeff, M.P., & Wallace, C.S. (1984). A general criterion for inductive inference. *Proceedings of the 6th European Conference on Artificial Intelligence*, Tim O'Shea (Ed.). Amsterdam: Elsevier.

Hamming, R.W. (1980). *Coding and information theory.* Englewood Cliffs, NJ: Prentice Hall.

Quinlan, J.R. & Rivest, R.L. (1989). Inferring decision trees using the minimum description length principle. *Information & Computation, 80,* 227-248.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81-106.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *Annals. of Statistics, 11,* 416-431.

Rissanen, J., & Langdon, G.G. (1981). Universal modeling and coding. *IEEE Transactions on Information Theory, IT-27,* 12-23.

Shannon, C.E., & Weaver, W. (1949). The mathematical theory of communication. Urbana: University of Illinois Press.

Wallace, C.S., & Boulton, D.M. (1968). An information measure for classification. *Computer Journal, 11,* 185-195.

Wallace, C.S., & Freeman, P.R. (1987). Estimation & inference by compact coding. *Journal of the Royal Statistical Society (B), 49,* 240-265.

Received March 21, 1991
Accepted February 5, 1992
Final Manuscript May 7, 1992