Statistics and Computing (1992) **2**, 63–73

Learning classification trees

WRAY BUNTINE

RIACS & NASA Ames Research Center, Mail Stop 269-2, Moffett Field, CA 94035, USA

Received January 1991 and accepted November 1991

Algorithms for learning classification trees have had successes in artificial intelligence and statistics over many years. This paper outlines how a tree learning algorithm can be derived using Bayesian statistics. This introduces Bayesian techniques for splitting, smoothing, and tree averaging. The splitting rule is similar to Quinlan's information gain, while smoothing and averaging replace pruning. Comparative experiments with reimplementations of a minimum encoding approach, C4 (Quinlan *et al.*, 1987) and CART (Breiman *et al.*, 1984), show that the full Bayesian algorithm can produce more accurate predictions than versions of these other approaches, though pays a computational price.

Keywords: Classification trees, Bayesian statistics

1. Introduction

A common inference task consists of making a discrete prediction about some object, given other details about the object. For instance, in financial credit assessment as discussed by Carter and Catlett (1987) we wish to decide whether to accept or reject a customer's application for a loan given particular personal information. This prediction problem is the basic task of many expert systems, and is referred to in artificial intelligence as the *classification problem* (where the prediction is referred to as the classification). The task is to learn a classifier given a *training sample* of classified examples. In credit assessment, classes are 'accept' and 'reject' (the credit application). A training sample in this case would be historical records of previous loan applications together with whether the loans went bad. This generic learning task is referred to as *supervised* learning in pattern recognition, and *induction* or *empirical learning* in machine learning (see, for instance, Quinlan, 1986). The statistical community uses techniques such as discriminant analysis and nearest neighbor methods, described, for instance, by Ripley (1987).

This prediction problem is often handled with *partitioning classifiers*. These classifiers split the example space into partitions; for instance, ID3 (Quinlan, 1986) and CART (Breiman *et al.*, 1984) use classification trees to partition the example space recursively, and CN2 (Clark and Niblett, 1989) and PVM (Weiss *et al.*, 1990) use disjunctive rules (which also partition a space, but not recursively in the manner of trees). Tree algorithms build trees such as the ones shown in Fig. 1. The medical tree shown on the left has the classes *hypo* (hypothyroid) and *not* (not hypothyroid) at the leaves. This tree is referred to as a *decision tree* because decisions about class membership are represented at the leaf nodes. Notice that the first test, $TSH > 200$, is a test on the real-valued attribute TSH. An example is classified using this tree by checking the current test and then falling down the appropriate branch until a leaf is reached, where it is classified. In typical problems involving noise, class probabilities are usually given at the leaf nodes instead of class decisions, forming a *class probability tree* (where each leaf node has a vector of class probabilities). A corresponding class probability tree is given in Fig. 1 on the right. The leaf nodes give predicted probabilities for the two classes. Notice that this tree is a representation for a conditional probability distribution of class given information higher in the tree. I will only be concerned with class probability trees in this paper since decision trees are a special case. Tree-based approaches have been pursued in many areas such as applied statistics, character recognition, and information theory for well over two decades. Perhaps the major classical statistics text in this area is Breiman *et al.* (1984), and a wide variety of methods and comparative studies exist in other areas (see, for example, Quinlan, 1988; Mingers, 1989b; Buntine, 1991b; Bahl *et al.*, 1989; Quinlan and Rivest, 1989; Crawford, 1989; and Chou, 1991).

The standard approach to building a class probability tree consists of several stages: growing, pruning, and sometimes smoothing or averaging. A tree is first *grown* to completion so that the tree partitions the training sample into terminal regions of all one class. This is usually done

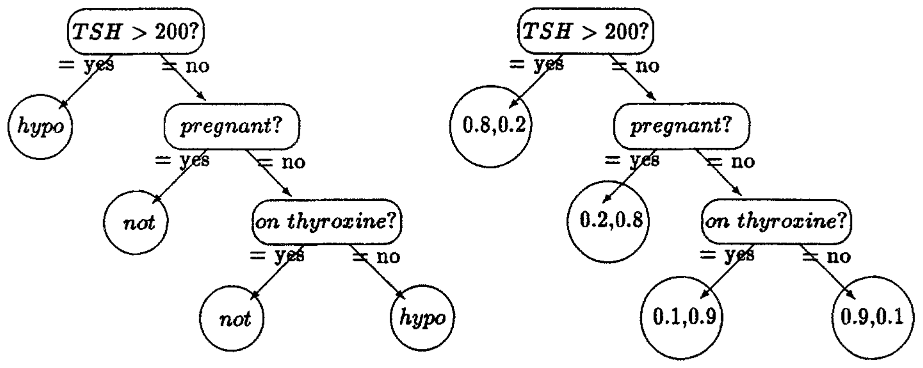

Fig. 1. *A decision tree and a class probability tree from the thyroid application*

from the root down using a *recursive partitioning* algorithm. Choose a test for the root node to create a tree of depth 1 and partition the training set among the new leaves just created. Now apply the same algorithm recursively to each of the leaves. The test is chosen at each stage using a greedy one-ply lookahead heuristic. Experience has shown that a tree so grown will suffer from over-fitting, in the sense that nodes near the bottom of the tree represent noise in the sample, and their removal can often increase predictive accuracy (for an introduction, see Quinlan, 1986). To help overcome this problem, a second process is subsequently employed to *prune* back the tree, for instance using resampling or hold-out methods (see, for example, Breiman *et al.*, 1984; or Crawford, 1989), approximate significance tests (Quinlan, 1988; or Mingers, 1989a), or minimum encoding (Quinlan and Rivest, 1989). The pruned tree may still have observed class probabilities at the leaves with zeros for some classes, an unrealistic situation when noisy data is being used. So *smoothing* techniques described by Bahl *et al.* (1989) and Chou (1991), explained later, are sometimes employed to make better class probability estimates. A final technique from Kwok and Carter (1990) is to build multiple trees and use the benefits of *averaging* to arrive at possibly more accurate class probability estimates.

This paper outlines a Bayesian approach to the problem of building trees that is related to the minimum encoding techniques of Wallace and Patrick (1991) and Rissanen (1989). These two encoding approaches are based on the idea of the 'most probable model', or its logarithmic counterpart, the minimum encoding. But the approaches here average over the best few models using two techniques; the simplest is a Bayesian variant of the smoothing technique of Bahl *et al.* (1989). A fuller discussion of the Bayesian methods presented here, including the treatment of real values and extensive comparative experiments, is given by Buntine (1991b). The current experiments are described in more detail by Buntine (1991a). Source code and manual for the implementations and reimplementations are available in some cases as the IND Tree Package by Buntine and Caruana (1991).

2. Theory

This section introduces the notation, the priors and the posteriors for the prediction task just described. This develops the theoretical tools on which the Bayesian tree learning methods are based. The section largely assumes the reader is familiar with the basics of Bayesian analysis, that is, the application of Bayes's theorem, the notion of subjective priors, etc (see, for instance, Press, 1989; or Lee, 1989). In what follows, the term 'prior' is an abbreviation for 'prior probability distribution'; likewise for 'posterior'. The prior and the posterior are both subjective Bayesian probabilities or measures of belief, and so do not correspond to measurable frequencies.

If the loss function to be used is minimum errors in prediction, we need to determine, given a new unclassified example, which class has maximum posterior probability of being true. In the Bayesian framework with trees used to represent knowledge about making predictions, this task involves determining posterior probabilities for different tree structures and class proportions, and then returning the posterior class probability vector for the new example based on posterior probability averaged over all possible trees. The mathematics of this process is outlined in this section.

2.1. *Basic framework*

To reason about posterior probabilities of class probability trees conditioned on a training sample, we need to separate out the discrete and the continuous components of a class probability tree. These need to be modeled by probability functions and probability density functions, respectively.

A class probability tree partitions the space of examples into disjoint subsets, each leaf corresponding to one such subset, and associates a conditional probability rule with each leaf. Denote the tree structure that defines the partition by T; this is determined by the branching structure of the tree together with the tests made at the internal nodes.

Suppose there are C mutually exclusive and exhausive classes, $d_1, \ldots, d_C$. The probabilistic rule associated with each leaf can be modeled as a conditional probabilty distribution. Suppose example x falls to leaf l in the tree structure T. Then the tree gives a vector of class probabilities $\phi_{j,l}$ for $j = 1, \ldots, C$, which give the proportion of examples at the leaf that have class d_j. A class probability tree then corresponds to a (discrete) tree structure T, together with the (continuous) matrix of class proportions $\boldsymbol{\Phi}_T = \{\phi_{j,l}: j = 1, \ldots, C, l \in \mathit{leaves}(T)\}$. If the choice of a test at a node requires selecting a 'cut-point', a real value, as does the test at the root of the tree in Fig. 1, then this framework needs to be modified. The 'cut-points' chosen are continuous, not discrete parameters, so their specification involves additional continuous parameters for the tree structure T. A tree T, $\boldsymbol{\Phi}_T$ therefore represents a conditional probability distribution for class c given example x of the form

$$\phi_{j,l} = P(c = d_j \mid x, T, \boldsymbol{\Phi}_T)$$

where example x falls to leaf l in the tree structure T.

For the learning problem, we are given a training sample $\boldsymbol{x}$, $\boldsymbol{c}$ consisting of N examples x_i with known classification given by class values c_i. Examples and their classes are assumed to be generated identically and independently. The distribution of a single classified example x, c can be specified by a probability distribution on the example, about which we have little concern, together with a conditional probability distribution on the class c given the example x, which corresponds to the class probability tree. Given a sample consisting of N examples $\boldsymbol{x}$ with known classification $\boldsymbol{c}$, we are interested in determining the posterior distribution of class probability trees given the training sample. This distribution can be found using Bayes's theorem:

$$\begin{aligned} P(T, \boldsymbol{\Phi}_T \mid \boldsymbol{x}, \boldsymbol{c}) &\propto P(T, \boldsymbol{\Phi}_T \mid \boldsymbol{x}) \prod_{i=1}^{N} P(c_i \mid x_i, T, \boldsymbol{\Phi}_T) \\ &= P(T, \boldsymbol{\Phi}_T \mid \boldsymbol{x}) \prod_{l \in \mathit{leaves}(T)} \prod_{j=1}^{C} \phi_{j,l}^{n_{j,l}} \end{aligned}$$

where $n_{j,l}$ is the number of examples of class d_j falling in the lth leaf of the tree structure T. The probability $P(T, \boldsymbol{\Phi}_T \mid \boldsymbol{x})$ I refer to as the *prior*, because even though it is conditioned on the unclassified examples $\boldsymbol{x}$, below I assume independence from these unclassified examples.

2.2. *Priors*

There are many different priors that could be used for trees. Entropic arguments like that of Rodriguez (1990) suggest that leaves should have more extreme probabilities (closer to 0 and 1) when the leaf is more infrequent (fewer examples occur at the leaves), for instance, lower down the tree. Trees tend to fracture the example space unnecessarily which Pagallo and Haussler (1990) refer to as the replication problem, because some tests higher in the tree are unnecessary for some (but not all) of the lower leaves. For this reason, we might expect class probabilities to be similar across different leaf nodes. Smoothness arguments also suggest that class probabilities for neighboring leaves in the tree might be more similar on average than for unrelated leaves. We therefore might use a hierarchical prior that relates class probabilities across leaves.

Rather than these, I use priors that have a simple conjugate (that is, the prior is the same functional form as the likelihood function; see Berger, 1985) and multiplicative form. Generic representation-independent techniques for forming priors, such as Rodriguez's entropic priors and smoothing priors described by Buntine and Weigend (1991), yield priors that lack the sample form I use. But although our motivation for choice of priors is expediency, I believe they are reasonable 'generic' priors for many circumstances.

Consider a prior consisting of a term for the tree structure and a term for the class proportions at each leaf given by

$$\begin{aligned} P(T, \boldsymbol{\Phi}_T \mid \boldsymbol{x}) &= P(T) P(\boldsymbol{\Phi}_T \mid T) \\ &= P(T) \prod_{l \in \mathit{leaves}(T)} \frac{1}{B_C(\alpha_1, \ldots, \alpha_C)} \prod_{j=1}^{C} \phi_{j,l}^{\alpha_j - 1} \end{aligned} \tag{1}$$

This assumes that different class probability vectors at the leaves are a priori independent. Although this contradicts our intuitions described above, it does not add any incorrect information a priori. B_C is the C-dimensional beta function defined by

$$B_C(x_1, \ldots, x_C) = \frac{\Pi_{j=1}^{C} \Gamma(x_j)}{\Gamma(\Sigma_{j=1}^{C} x_j)}$$

and Γ is the gamma function. Each term indexed by l in Equation 1 is a Dirichlet distribution (a C-dimensional variant of the two-dimensional beta distribution). Let $\alpha_0 = \Sigma_{j=1}^{C} \alpha_j$. The Dirichlet terms ensure that the class probability vectors at each leaf l center about a common mean

$$\left(\frac{\alpha_1}{\alpha_0}, \ldots, \frac{\alpha_C}{\alpha_0} \right)$$

with standard deviation proportional to $(\alpha_0 + 1)^{-1/2}$. I take an empirical Bayes approach (see Berger, 1985) and set the common mean to the observed common mean in the training sample. Because of the sample size, this should be close to the population base-rate. The 'prior weight' parameter α_0 is then set according to how much we expect a priori the class probabilities at leaves to vary from the base-rate class probabilities. A value of $\alpha_0 = 1$ means we expect class probabilities to differ strongly from the base-rate, and a value of $\alpha_0 = C$ means we expect mild difference.

I use several choices for the prior on the tree structure T, $P(T)$. Each of these priors is multiplicative on the nodes in the tree, so the posterior, given later, is also multiplicative on the nodes in the tree. Also, the priors I–III presented are not normalized as their later use does not require it, and prior I is a special case of prior II.

I. Each tree structure is equally likely, $P(T)$ is constant.

II. Give a slight preference to simpler trees, such as $P(T) \propto \omega^{|nodes(T)|}$, for $\omega < 1$. This means every node in the tree makes the tree a factor of ω less likely a priori.

III. The tree shapes (tree structure minus choice of tests at internal nodes) are equally likely.

$$P(T) \propto \prod_{n \in nodes(T) - leaves(T)} \frac{1}{\#\, possible\ tests\ at\ n}$$

This means we have a uniform prior on the number of tests that will be made on an example to determine its class probability.

IV. Tree structures are coded using bits for 'node', 'leaf', and 'choice of test'. This is a combination of type II and III priors together with 'encoding' of cut-points. (For details, see Rissanen, 1989, p. 165; or Wallace and Patrick, 1991).

The priors have been given in increasing strength, in the sense that type IV priors represent an extreme preference for simpler trees, but type I priors represent no such preference. In medical data, for instance, where many of the attributes supplied for each example may well be noise, we would expect the stronger priors to be more reasonable. In problems like the classic faulty LED problem of Breiman *et al.* (1984), we would expect all faulty LED indicators to be somewhat informative about the actual digit being represented, so large trees seem reasonable and the type I prior should be used.

Clearly, many variations on these priors are possible without departing from the basic conjugate and multiplicative form. Certain attributes could be penalized more than others if it is believed a priori that those attributes are not as effective in determining class. The prior weight α_0 could vary from node to node, for instance becoming smaller lower down the tree. Our claim, however, is that the priors described above are an adequate family of mildly informative tree priors for the purposes of demonstrating a Bayesian approach to learning tree classifiers.

2.3. *Posteriors*

Using standard properties of the Dirichlet distributions, given by Buntine (1991b, Section 2.5), posteriors conditioned on the training sample can now be computed as follows:

$$\begin{aligned} P(T, \Phi_T \mid x, c) &= P(T \mid x, c) \cdot P(\Phi_T \mid x, c, T) \\ P(T \mid x, c) &\propto P(T) \prod_{l \in leaves(T)} \frac{B_C(n_{1,l} + \alpha_1, \ldots, n_{C,l} + \alpha_C)}{B_C(\alpha_1, \ldots, \alpha_C)} \\ P(\Phi_T \mid x, c, T) &\propto \prod_{l \in leaves(T)} \frac{1}{B_C(\alpha_1, \ldots, \alpha_C)} \prod_{j=1}^{C} \phi_{j,l}^{n_{j,l} + \alpha_j - 1} \end{aligned} \tag{2}$$

One important property of these posteriors is that they are multiplicative on the nodes in the tree whenever the prior is also multiplicative on the nodes. This means posteriors for a collection of similar trees can be efficiently added together using an extended distributive law as described below in Section 3.2. A second important property is that the discrete space of tree structures is combinatoric, and only partially ordered. This means for smaller data sets we might expect to have many different shaped trees with a similar high posterior. In contrast, consider the space of polynomials of a single variable. This is a linearly ordered discrete space, so we can expect polynomials with a high posterior to be polynomials of a similar order.

Posterior expected values and variances for the proportions Φ_T can be deduced from the posterior using standard properties of the Dirichlet distribution. I use the notation $E_{x \mid y}(z(x, y))$ to denote the expected value of $z(x, y)$ according to the distribution for x conditioned on knowing y. For instance, the posterior expected class proportions given a particular tree structure are:

$$E_{\Phi_T \mid x, c, T}(\phi_{j,l}) = \frac{n_{j,l} + \alpha_j}{n_{.,l} + \alpha_0} \tag{3}$$

where $n_{.,l} = \Sigma_{j=1}^{C} n_{j,l}$ and $\alpha_0 = \Sigma_{j=1}^{C} \alpha_j$.

The log-posterior for the tree structure can be approximated when the sample size at each leaf ($n_{.,l}$) is large using Sterling's approximation:

$$-\log P(T \mid x, c) \simeq -\log P(T) + NI(C \mid T) + constant \tag{4}$$

where the dominant term $I(C \mid T)$ represents the expected information gained about class due to making the tests implicit in the tree structure T. That is:

$$I(C \mid T) \equiv \sum_{l \in leaves(T)} \frac{n_{.,l}}{N} I\left(\frac{n_{1,l} + \alpha_1}{n_{.,l} + \alpha_0}, \ldots, \frac{n_{C,l} + \alpha_C}{n_{.,l} + \alpha_0}\right).$$

The function I used here is the standard information or entropy function over discrete probability distributions. The information measure $I(C \mid T)$ when applied to a tree of depth 1 is used by Quinlan (1986) as a splitting rule when growing trees. If some leaves have small numbers of examples, then Approximation 4 is poor, and the beta function really needs to be used. The beta function then has the effect of discounting leaves with small example numbers.

3. Methods

To implement a full Bayesian approach, we need to consider averaging over possible models, as, for instance, approximated by Kwok and Carter (1990), or done by Stewart (1987) using Monte Carlo methods. This section introduces methods closer in spirit to Henrion (1990) who collects the dominant terms in the posterior sum. Fuller details of the methods described below are given by Buntine (1991b).

Intuitively, this full Bayesian approach involves averaging all those trees that seem a reasonable explanation of the classifications in the training sample, and then predicting the class of a new example based on the weighted predictions of the reasonable trees. This estimates the posterior probability conditioned on the training sample that a new example x will have class c by the formula:

$$E_{T,\boldsymbol{\Phi}_T|\boldsymbol{x},\boldsymbol{c}}(P(c=d_j|x,T,\boldsymbol{\Phi}_T)) = \frac{\Sigma_{T,\,l=leaf(x,T)} P(T,\boldsymbol{x},\boldsymbol{c}) \dfrac{n_{j,l}+\alpha_j}{n_{\cdot,l}+\alpha_0}}{\Sigma_T P(T,\boldsymbol{x},\boldsymbol{c})} \tag{5}$$

where the summations are over all possible tree structures, and *leaf*(x, T) denotes the unique leaf in the tree T to which the example x falls. Notice that the denominator of the right-hand side of Equation 5 can be simplified. However, when approximating the summations with a reduced set of tree structures as done below, this full form is required to normalize the result.

Given the combinatoric number of tree structures, such a calculation is not feasible. There are three computationally reasonable approximations to this general formula that can be made by restricting the summations to a reduced subset of tree structures. The first corresponds to a maximum a posteriori approximation, and the second and third, described below, collect more dominant trees from the a posteriori sum. To collect these dominant trees, however, we first have to find high a posteriori trees. The growing method presented in Section 3.1 describes how.

3.1. *Tree growing*

Formula 2 suggests a heuristic for growing a tree in the standard recursive partitioning algorithm described in Section 1. When expanding the (single) current node, for each possible test grow a tree of depth 1 at that node by extending it one more level. Then choose the new test yielding a tree structure with the maximum posterior probability. Because the posterior is multiplicative, we only need look at that component of the posterior contributed by the new test and its leaves. The one-ply lookahead heuristic for evaluating a test then becomes:

$$Quality_1(test) = P(test) \times \prod_{l \in outcomes(test)} \frac{B_C(n_{1,l}+\alpha_1,\ldots,n_{C,l}+\alpha_C)}{B_C(\alpha_1,\ldots,\alpha_C)} \tag{6}$$

where $P(test)$ is the contribution to the tree prior, *outcomes*$(test)$ is the set of test outcomes, and $n_{j,l}$ is the number of examples in the training sample at the current node with class j and having test outcome l. So the $n_{j,l}$ are also a function of the test. The subscript 1 is there in $Quality_1(test)$ to remind us that this is a one-ply lookahead heuristic. Computation should be done in log-space to avoid underflow. A test should be chosen to maximize Formula 6.

If the test being evaluated contains a cut-point, as does the test at the root of the tree in Fig. 1, then this too should be averaged over. Formula 6 in this case represents an evaluation of the test conditioned on knowing the cut-point. Since we have the freedom to choose this as well, so we should calculate the expected value of Formula 6 across the full range of cut-points. Suppose we assume all cut-points are a priori equally likely between a minimum and maximum value, then for a real-valued attribute R, the quality of a cut-point test on R, denoted *cut-point*(R) is given by

$$Quality_1(cut\text{-}point(R)) = \frac{1}{max(R)-min(R)} \times \int_{cut=min(R)}^{max(R)} Quality_1(R<cut)\, \mathrm{d}cut \tag{7}$$

This test adds a penalty factor, given by $Quality_1(cut\text{-}point(R))/Quality_1(R<best\text{-}cut) \ll 1.0$, to the quality of the best cut, *best-cut*. This penalty has the same effect as the 'cost of encoding the cut-point', added by Quinlan and Rivest (1989) in their minimum encoding approach. While this evaluates the average quality of the cut-point test, we also need to select a cut-point. I use the cut that maximizes the quality test $Quality_1(R<cut)$.

Experiments show the quality heuristic of Equation 6 is very similar in performance to the information gain measure of Quinlan (1986) and to the GINI measure of CART (Breiman *et al.*, 1984). Approximation 4 explains why. The quality measure has the distinct advantage, however, of returning a measure of quality that is a probability. This can be used to advantage when growing trees from a very large training sample, growing trees incrementally, or after a stopping or pre-pruning rule.

When determining the test to make at a node using a one-ply lookahead heuristic, we need to do $O(AN)$ operations, or $O(AN \log N)$ if a sort is involved, where N is the size of the sample at that node, and A is the number of potential tests. To reduce computation for very large N, we could evaluate the tests on a subset of the sample (i.e. reduce N in the computation), as suggested by Breiman *et al.* (1984) and Catlett (1991). Because the measure of quality is in units of probability, one can readily determine if one test is significantly better than another according to the measure simply by taking their ratio. This can be used to determine if evaluation on the current subsample is sufficient, or if we need to view a larger subsample.

A related problem occurs when growing trees incrementally. In this regime, a tree needs to be updated given some additions to the training sample. Crawford (1989) shows that if we take the naive approach as done by Utgoff (1989) and attempt to update the current tree so the 'best' split according to the updated sample is taken at each node, the algorithm suffers from repeated restructuring. This occurs because the best split at a node vacillates while the sample at the node is still small. To overcome this problem Crawford suggests allowing restructuring only if some other test is currently significantly better than the current test at the node. We also need a test to determine whether the test at a node should be fixed and not changed thereafter, despite new data. Crawford uses a relatively expensive resampling approach to determine significance of splits, but the probabilistic quality measure of Formula 6 can be used instead without further modification.

The quality heuristic can also be improved using an N-ply lookahead beam search instead of the one-ply. In this, I use the one-ply quality test to obtain a few good tests to make up the search 'beam' at this level. I currently use a beam (search width) of 4. I then expand these tests, do an $(N-1)$-ply lookahead beam search at their children to find the best tests for the children, and finally propagate the quality of the children to calculate a quality measure corresponding to Equation 2 for the best subtree grown. This means that, at the cost of computation, we can do more extensive searches.

3.2. *Tree smoothing*

The simplest pruning approach is to choose the maximum a posteriori tree. Of all those tree structures obtained by pruning back the complete grown tree, pick the tree structure with maximum posterior probability, $P(T \mid \boldsymbol{x}, \boldsymbol{c})$. This pruning approach was tried with a range of different priors and the approach was sometimes better in performance than Quinlan's C4 or Breiman *et al.*'s CART, and sometimes worse. With careful choice of prior, this 'most probable model' approach was often better; however, it was unduly sensitive to the choice of prior. This suggest a more thorough Bayesian approach is needed.

The first approximation to the sum of Equation 5 I call *Bayesian smoothing*. The standard approach for classifying an example using a class probability tree is to send the example down to a leaf and then return the class probability at the leaf. In the smoothing approach, I also average all the class probability vectors encountered at interior nodes along the way (see Bahl *et al.*, 1989, p. 1005). Given a particular tree structure T', presumably grown using the algorithm described in Section 3.1, consider the space of tree structures, *pruned*(T'), obtained by pruning the tree structure T' in all possible ways. If we restrict the summation in Equation 5 to this space and the posterior on the tree structure is a multiplicative function over nodes in the tree, then the sum can be recursively calculated using a grand version of the distributive law. The sum is computable in a number of steps linear in the size of the tree. The sum takes the form of an average calculated along the branch traversed by the new example.

$$E_{T,\Phi_T \mid x, c}(P(c = d_j \mid x, T, \Phi_T))$$

$$\simeq \frac{\sum_{T \in pruned(T'),\, l = leaf(x,T)} \Pr(T, \boldsymbol{x}, \boldsymbol{c}) \dfrac{n_{j,l} + \alpha_j}{n_{\cdot,l} + \alpha_0}}{\sum_{T \in pruned(T')} \Pr(T, \boldsymbol{x}, \boldsymbol{c})}$$

$$= \sum_{n \in traversed(x,\, T')} P(n \text{ is } leaf \mid \boldsymbol{x}, \boldsymbol{c}, pruning \text{ of } T') \times \frac{n_{j,n} + \alpha_j}{n_{\cdot,n} + \alpha_0} \tag{8}$$

where *traversed*(x, T') is the set of nodes on the path traversed by the example x as it falls to a leaf, and $\Pr(n$ *is leaf* $\mid \boldsymbol{x}, \boldsymbol{c}$, *pruning of* $T')$ is the posterior probability that the node n in the tree T' will be pruned back to a leaf given that the 'true' tree is a pruned subtree of T'. It is given by

$$P(n \text{ is } leaf \mid \boldsymbol{x}, \boldsymbol{c}, pruning \text{ of } T')$$

$$= CP(leaf(n), \boldsymbol{x}, \boldsymbol{c}) / SP(T', \boldsymbol{x}, \boldsymbol{c})$$

$$\times \prod_{O \in ancestors(T',n)} CP(node(O)) \prod_{\substack{B \in children(O) \\ B \neq child(O,x)}} SP(B, \boldsymbol{x}, \boldsymbol{c})$$

where *ancestors*(T', n) is the set of ancestors of the node n in the tree T', *child*(O, x) is the child tree of the node O taken by the example x during traversal, *children*(O) is the set of children trees of the node O,

$$SP(T, \boldsymbol{x}, \boldsymbol{c}) = \sum_{ST \in pruned(T)} P(ST, \boldsymbol{x}, \boldsymbol{c}) = CP(leaf(T), \boldsymbol{x}, \boldsymbol{c})$$

$$+ 1_{T \text{ is not } leaf} CP(node(T)) \prod_{B \in branches(T)} SP(B, \boldsymbol{x}, \boldsymbol{c}) \tag{9}$$

and $CP(node(T))$ is the component of the tree prior due to the internal node at the root of T, for instance 1 for type I priors and $1/\#$*possible tests at node*(T) for type III priors. For $n_{1,l}, \ldots, n_{C,l}$ the class counts at the node T,

$$CP(leaf(T), \boldsymbol{x}, \boldsymbol{c}) = CP(leaf(T)) \frac{B_C(n_{1,l} + \alpha_1, \ldots, n_{C,l} + \alpha_C)}{B_C(\alpha_1, \ldots, \alpha_C)}$$

where $CP(leaf(T))$ is the multiplicative component of the tree prior due to the leaf T, for instance 1 for type I priors and ω for type II priors. Cut-points are currently handled by multiplying in the penalty factor described in Section 3.1 with *CP(node(T))*.

For example, consider the tree given in Fig. 2. This is grown from data about voting in the US Congress. The numbers at the nodes represent class counts of Democrats and Republicans, respectively. To smooth this counts tree, we first compute the class probabilities for each node, and compute a node probability indicating how strongly we believe that node *n* gives appropriate class probabilities, $P(n \text{ is leaf} \mid x, c, \text{pruning of } T')$. This was done with $\alpha_1 = \alpha_2 = 0.5$ and a type II tree prior with $\omega = 0.5$, intended to bias against larger trees. The intermediate tree is given in Fig. 3. The probability in brackets at each node represents the node probability. Notice these sum to 1.0 along any branch. The two probabilities below represent the class probabilities for that node for Democrats and Republicans, respectively. This intermediate tree allows smoothing as follows. Suppose we have a politician voting 'yes' on *el-salv-aid*, *educ-spend* and *mx-missile*. Then the probability the politician is Republican is given by running down the rightmost branch and computing the weighted sum of class probabilities:

$$0.000 \times 0.35 + 0.355 \times 0.76 + 0.348 \times 0.88 + 0.296 \times 0.75 = 0.80$$

The final tree after performing this smoothing process is given in Fig. 4. Notice the difference in probabilities for

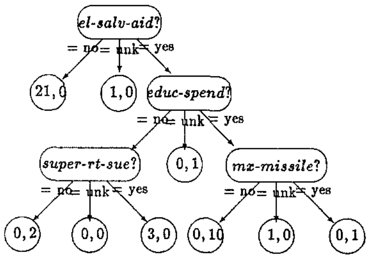

Fig. 2. *A class counts tree from the congressional voting application*

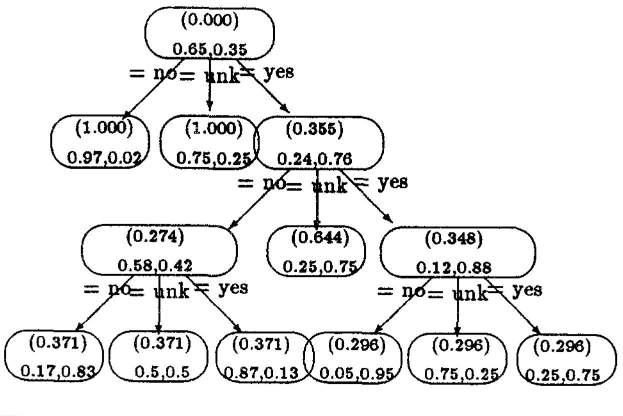

Fig. 3. *The intermediate calculation tree*

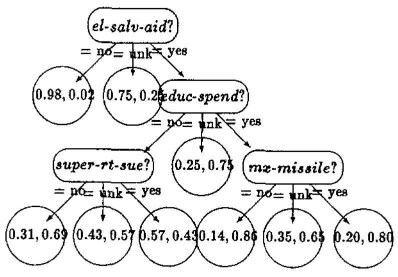

Fig. 4. *The averaged class probability tree*

the leaf nodes of the intermediate class probability tree and the averaged class probability tree. In particular, notice that for the averaged tree the bottom right test on *mx-missile* has all its leaf nodes predict Republican. Rather than pruning these three leaf nodes we can keep them separate because the probabilities for the middle leaf are quite different from the probabilities for the other leaves. Of course, since the class counts are all quite small, a change in the prior parameters α_j and ω could change the final class probabilities quite a bit.

In experiments with smoothing, sometimes nodes will make so little contribution to the final averaged probabilities that they can be pruned without much affecting class probabilities of the resultant tree. For instance, this would happen if the cluster of leaves at the bottom right of Fig. 3 under the test *mx-missile* all had leaf probabilities of 0.001 instead of 0.296. This means the contribution to the sum in Equation 8 by a traversed node *n* and all its descendants will be so small that they will have no effect on the sum. In this case nodes *n* and below can be pruned.

Experiments reported below showed smoothing often significantly improved class probability estimates for a class probability tree (for instance, as measured using the half-Brier score), and sometimes made no significant difference. This happened regardless of the pruning and growing approach used to construct the original tree. In some cases smoothing is an adequate replacement for tree pruning compared with the standard pruning approaches such as pessimistic pruning or cost-complexity pruning. However, for really noisy data using a weak prior, this form of pruning was not strong enough, whereas, for strongly structured data with a strong prior, the pruning was too severe due to the choice of prior. The smoothing approach gives predictions still quite sensitive to the prior, although it was generally better than or at least as good as finding the most probable model.

3.3. *Option trees and averaging*

The second approximation to the sum of Equation 5 involves searching for and compactly storing the most dominant (i.e. highest posterior) tree structures in the sum. The approach involves building *option trees* and then

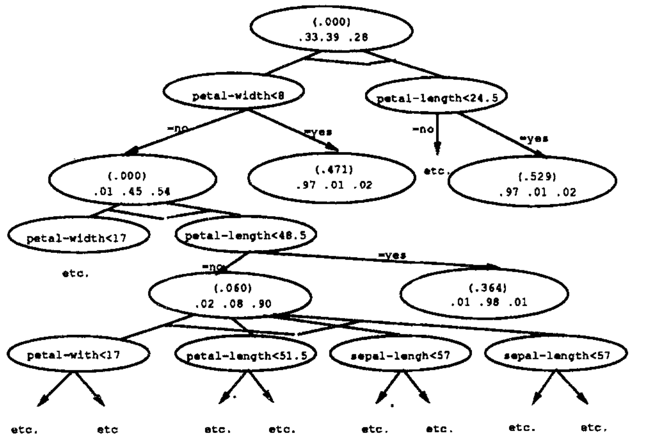

Fig. 5. *A class probability option tree from the iris application*

doing *Bayesian averaging* on these trees. Option trees are a generalization of the standard tree where options are included at each point. At each interior node, instead of there being a single test and subtrees for its outcomes, there are several optional tests with their respective subtrees. The resultant structure looks like an and–or tree. This is a compact way of representing many different trees that share common prefixes.

A class probability option tree built from Fisher's iris data is represented in Fig. 5. These trees have leaf nodes and test nodes as before, but optional test nodes are sometimes XORed together in a cluster. For instance, the very top node of the option tree XORs together two test nodes, *petal-width* < 8 and *petal-length* < 24.5. These two tests in the small ovals are referred to as *options* because, when selecting a single tree, we are to choose exactly one. Each node is labeled in brackets with the probability with which the node should be a leaf, which corresponds to the first probability in Equation 8. These determine the probability of selecting any single tree. Each node is also labeled with the estimated class probabilities at the node, used when averaging. The cluster of two tests at the top is referred to as a node of degree 2. So for the top node of degree 2, we should treat it as a leaf with probability 0.0 and otherwise choose either the test *petal-width* < 8 or the test *petal-length* < 24.5. Both options have subtrees with high probability leaves, so these two optional trees are about as good as each other. When parts of the option tree are not included in the figure, the word 'etc.' appears. The bottom cluster of options is a node of degree 4. This has its subtrees excluded from the figure. That this bottom cluster and its children contain optional tests on all four attributes indicates that this cluster gives little indication as to which test is more appropriate, or if any test is appropriate at all.

Classification on the resultant structure is done using Equation 5 in a similar style to Bayesian smoothing. The same formulae apply accept that Equation 9 repeats the second term for every option at the node. Because this no longer involves dealing with a single tree, I refer to the process of growing and smoothing as *tree averaging*.

The current method for growing option trees is primitive yet adequate for demonstration purposes. Option trees are grown using an N-ply lookahead as described in Section 3.1. Rather than only growing the best test as determined by the lookahead, the best few tests are grown as optional tests at the node. I currently use one-ply or two-ply lookahead and allow a maximum of four optional tests at each node, retaining only those within a factor of 0.005 of the best test. A depth bound is used to stop growing the tree after a fixed depth, and this is chosen a priori, although it sometimes had to be decreased due to a memory or time overrun. Note that these parameters affect the search, and more search usually leads to better prediction accuracy at the expense of a larger option tree. For nodes that have large counts, usually only one test is expanded because all others are insignificant according to the quality measure. With smaller samples, or in domains where trees are a poor representation (such as noisy DNF concepts) many tests may be almost as good according to the quality measure so many options will be expanded. This means option trees tend to have more options at the lower nodes where more uncertainty lies.

4. Comparison

4.1. *Experimental setup*

Comparative trials were run with the Bayesian algorithms discussed above and reimplementations of CART, C4, and a generic minimum encoding method. The versions of CART and C4 were reimplementations by me, and are known to perform comparably to the original algorithms on the data sets used. With the CART reimplementation, cost-complexity pruning was done with tenfold cross-validation using the 0-SE rule. Breiman *et al.* suggest this gives the most accurate predictions, and this was confirmed experimentally. For the Bayesian algorithms, I used either type I or type II priors with $\omega = 0.5$, depending on whether I believed there were many irrelevant attributes. The prior weight parameter α_0 was set depending on how strongly I believed class probabilities would vary from the base-rate. These prior choices were fixed before running the algorithms. To standardize the experiments, all methods used the tree growing method of Section 3.1. This behaves similarly to the standard GINI and information-gain criteria on binary splits.

Data sets came from different real and simulated domains, and have a variety of characteristics. They include Quinlan's (1988) hypothyroid and XD6 data, the CART digital LED problem, medical domains reported by Cestnik, Kononenko and Bratko (1987), pole balancing data from human subjects collected by Michie, Bain and Hayes-Michie (1990), and a variety of other data sets from the Irvine Machine Learning Database such as 'glass', 'voting records', 'hepatitis' and 'mushrooms'. The 'voting' data has had the attribute 'physician-fee-freeze' deleted, as recommended by Michie. These are available via ftp at ics.uci.edu in "/pub".

For the LED data, $\alpha_0 = 1$ and the type I tree prior was used. This was because I believe all attributes are relevant and I expect a high accuracy. The type II tree prior was used for the pole balancing, voting and hepatitis domains because I believe attributes to be only partly relevant. For pole balancing I was inclined to use a type I prior, however it turned out not to make much difference. For the hepatitis domain, $\alpha_0 = 2$ because I expected class probabilities to lie around the population base-rate, rather than to be near 0 or 1. For many of these domains depth bounds on option trees were set at about 6.

Data sets were divided into training/test pairs, a classifier was built on the training sample and the accuracy, predicted accuracy, and half-Brier score taken on the test sample. The half-Brier score is an approximate measure of the quality of class probability estimates, similar to mean-square error, where smaller is better. This was done for ten random trials of the training/test pair, and significance of difference between two algorithms was checked using the paired t-test. Several different training-set sizes were also tried for each data set to test small and large sample properties of the algorithms.

Algorithms are part of the IND Tree Package and options used on Version 1.1; more details of the data sets, and acknowledgements to the sources are given by Buntine (1991a).

4.2. *Results*

A somewhat random selection of the results is presented in Table 1. The MSE column refers to half-Brier score. The 'Bayes trees' methods corresponds to one-ply lookahead growing with Bayesian smoothing.

Table 1. *Performance statistics*

Data	Method	Set size (train + test)	Timing (train + test)	Error ± std.dev.	MSE
LED	CART-like	100 + 2900	1.2 s + 1.0 s	35.7 ± 4.0	0.57
LED	C4-early	100 + 2900	0.8 s + 1.0 s	36.1 ± 4.6	0.59
LED	Bayes trees	100 + 2900	0.8 s + 1.8 s	35.5 ± 2.6	0.55
LED	1-ply option trees	100 + 2900	14.0 s + 62.2 s	33.9 ± 2.5	0.51
LED	2-ply option trees	100 + 2900	15.3 s + 71.4 s	33.8 ± 2.5	0.51
LED	CART-like	900 + 2100	2.3 s + 0.7 s	28.8 ± 0.6	0.45
LED	C4-early	900 + 2100	1.0 s + 0.7 s	29.3 ± 0.5	0.46
LED	2-ply option trees	900 + 2100	12.4 s + 12.0 s	28.0 ± 0.6	0.41
pole	CART-like	200 + 1647	15.0 s + 3.8 s	15.7 ± 1.0	0.26
pole	C4-early	200 + 1647	3.4 s + 3.4 s	15.6 ± 1.6	0.27
pole	1-ply option trees	200 + 1647	171.2 s + 174.8 s	15.2 ± 0.6	0.22
pole	CART-like	1200 + 647	38.0 s + 1.0 s	12.2 ± 1.7	0.21
pole	MDL-like	1200 + 647	4.0 s + 0.7 s	13.0 ± 1.3	0.20
pole	Bayes trees	1200 + 647	3.9 s + 0.8 s	12.3 ± 0.9	0.18
pole	1-ply option trees	1200 + 647	269.8 s + 18.3 s	10.6 ± 0.8	0.16
glass	CART-like	100 + 114	4.8 s + 0.3 s	37.3 ± 4.7	0.60
glass	C4-early	100 + 114	1.5 s + 0.3 s	36.2 ± 4.3	0.59
glass	Bayes tree	100 + 114	3.0 s + 0.3 s	36.5 ± 5.8	0.56
glass	2-ply option trees	100 + 114	469.9 s + 102.1 s	30.5 ± 6.0	0.43
voting	CART-like	200 + 235	1.4 s + 0.1 s	12.3 ± 1.7	0.22
voting	C4-early	200 + 235	0.9 s + 0.1 s	12.9 ± 1.5	0.22
voting	MDL-like	200 + 235	1.0 s + 0.2 s	12.9 ± 1.4	0.21
voting	1-ply option trees	200 + 235	162.5 s + 18.0 s	10.4 ± 1.5	0.16
hepatitis	CART-test	75 + 70	4.2 s + 0.1 s	19.8 ± 3.7	0.35
hepatitis	Bayes tree	75 + 70	1.5 s + 0.1 s	23.1 ± 4.9	0.32
hepatitis	2-ply option trees	75 + 70	131.0 s + 23.1 s	18.8 ± 3.6	0.26

In general, Bayesian option trees and averaging yielded superior prediction accuracy. It was always competitive and usually superior to the other algorithms. In several cases it was pairwise significantly better than each of the non-Bayesian approaches at the 5% level. Bayesian option trees and averaging with a one-ply and two-ply lookahead during growing yielded improvement in predictive accuracy averaged over 20 trials as often as high as 2–3%, sometimes more. Bayesian smoothing on either trees or option trees also yielded superior half-Brier scores. This is highly significant in most cases, even, for instance, when the prediction accuracy was worse. These results were consistent across most data sets and sizes, including those not shown here.

Bayesian option trees and averaging are also quite resilient to the setting of the prior parameters. Use of either type I or II tree prior often made little effect. Simple smoothing by contrast was quite sensitive to the prior. The effects of averaging are clearly important.

4.3. *Discussion*

With this simple experimental setup, it is difficult to say with conviction that any one algorithm is 'better' than the others, since there are several dimensions on which learning algorithms can be compared, and there are combinations of algorithms which were not tried. Prediction accuracy and half-Brier scores of the Bayesian methods are impressive, however.

Several factors contribute here: The option trees are more than just a single decision tree, they effectively involve an extension of the model space, so we are not comparing like with like. The growing of option trees sometimes involved an extra order of magnitude in time and space, partly perhaps because of the primitive search used. Option trees do not have the comprehensibility of normal trees, although I believe this could be arranged with some post-processing.

While option trees were often significantly better in accuracy by several percent, it is unclear how much of this is due to the smoothing/averaging process and how much is due to the improved multi-ply lookahead search during growing. Initial experiments combining multi-ply lookahead growing and CART-style cost-complexity pruning produced erratic results, and it is unclear why.

A final point of comparison is the parameters available when driving the algorithms. CART and C4 have default settings for their parameters. With CART, heavy pruning can be achieved using the 1-SE rule rather than the 0-SE rule. The number of partitions to use in cross-validation cost-complexity pruning can also be changed, but the effect of this is unclear, especially since leaving-one-out cross-validation cost-complexity pruning gives poor predictive accuracy. The minimum encoding approaches are (according to the purist) free of parameters. However, these approaches often strongly overprune, so Quinlan and Rivest (1989) introduce a parameter that allows lighter pruning. So all approaches, Bayesian and non-Bayesian, have parameters that allow more or less pruning that can be chosen depending on the amount of structure believed to exist in the data. In the fuller Bayesian approach with option trees and Bayesian averaging, choices available also allow greater search growing and fuller elaboration of the available optional trees. These parameters have the useful property that predictive accuracy (or some other utility measure) and computational expense are, on average, monotonic in the value of the parameter. The parameter setting allows improved predictive accuracy at computational expense.

5. Conclusion

Bayesian algorithms for learning class probability trees were presented and compared empirically with reimplementations of existing approaches like CART (Breiman *et al.*, 1984), C4 (Quinlan, 1988) and minimum encoding approaches. The Bayesian option trees and averaging algorithm gave significantly better accuracy and half-Brier score on predictions for a set of learning problems, but at computational expense. Bear in mind the Bayesian algorithms had settings of mild prior parameters made and undertook considerably more search, whereas the other algorithms were not tuned in any such way.

First, this work has a number of implications for Bayesian learning. Simple maximum posterior methods and minimum encoding methods (which here would choose the single maximum posterior tree) may not perform well in combinatorial discrete spaces like trees if the prior is not well matched to the problem. Considerable improvement can be obtained by averaging over multiple high posterior models. With trees and a multiplicative posterior, efficient averaging over multiple models is possible. Standard computational techniques for performing averaging such as importance sampling and Gibbs sampling are therefore avoided. More sophisticated priors could help here, but it is surely just as important to consider more versatile classification models such as the decision trellises suggested by Chou (1991).

Second, the Bayesian methods derived here corresponded to a variety of subtasks previously done by a collection of disparate *ad hoc* approaches honed through experience. The splitting rule derived here suggested improvements such as multi-ply lookahead search, penalty factors for cut-points, and a modification for doing learning in an incremental rather than a batch mode. A comparison with previous pruning and smoothing methods is difficult because the derived Bayesian methods are highly parametric, although cost-complexity pruning is in some ways comparable with use of the type II prior. Cross-validation is difficult to interpret from a Bayesian perspective.

More research is needed on these Bayesian methods. Multi-ply lookahead and smoothing could be combined with CART-style methods. It is unknown how much smoothing, option trees and multi-ply lookahead each contribute to the observed gain in prediction accuracy and half-Brier score. Further priors need to be developed. For instance, the current tree structure priors are difficult to conceptualize, and the whole Bayesian framework becomes dubious when priors are not somehow 'meaningful' to the user. More advanced any-time best-first searches could be developed for option trees, and an importance sampling approach might also compare favorably.

Acknowledgements

Thanks to Peter Cheeseman, Stuart Crawford and Robin Hanson for their assistance and to the Turing Institute and Brian Ripley at the University of Strathclyde who sponsored me while much of this research was taking shape. I am particularly indebted to Ross Quinlan, whose grasp of the tree field and insistence on experimental evaluation helped me enormously during the thesis development.

References

Bahl, L., Brown, P., de Souza, P. and Mercer, R. (1989) A tree-based language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **37**, 1001–1008.

Berger, J. O. (1985) *Statistical Decision Theory and Bayesian Analysis*, Springer-Verlag, New York.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984) *Classification and Regression Trees*, Wadsworth, Belmont.

Buntine, W. (1991a) Some experiments with learning classification trees. Technical report, NASA Ames Research Center. In preparation.

Buntine, W. (1991b) A theory of learning classification rules. PhD thesis, University of Technology, Sydney.

Buntine, W. and Caruana, R. (1991) Introduction to IND and recursive partitioning. Technical Report FIA-91-28, RIACS and NASA Ames Research Center, Moffett Field, CA.

Buntine, W. and Weigend, A. (1991) Bayesian back-propagation. *Complex Systems*, **5**, 603–643.

Carter, C. and Catlett, J. (1987) Assessing credit card applications using machine learning. *IEEE Expert*, **2**, 71–79.

Catlett, J. (1991) Megainduction: machine learning on very large databases. PhD thesis, University of Sydney.

Cestnik, B., Kononenko, I. and Bratko, I. (1987) ASSISTANT86: A knowledge-elicitation tool for sophisticated users, in *Progress in Machine Learning: Proceedings of EWSL-87*, Bratko, I. and Lavrač, N. (eds), Sigma Press, Wilmslow, pp. 31–45.

Chou, P. (1991) Optimal partitioning for classification and regression trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**.

Clark, P. and Niblett, T. (1989) The CN2 induction algorithm. *Machine Learning*, **3**, 261–283.

Crawford, S. (1989) Extensions to the CART algorithm. *International Journal of Man–Machine Studies*, **31**, 197–217.

Henrion, M. (1990) Towards efficient inference in multiply connected belief networks, in *Influence Diagrams, Belief Nete and Decision Analysis*, Oliver, R. and Smith, J. (eds), Wiley, New York, pp. 385–407.

Kwok, S. and Carter, C. (1990) Multiple decision trees, in *Uncertainty in Artificial Intelligence 4*, Schachter, R., Levitt, T., Kanal, L. and Lemmer, J. (eds), North-Holland, Amsterdam.

Lee, P. (1989) *Bayesian Statistics: An Introduction*, Oxford University Press, New York.

Michie, D., Bain, M. and Hayes-Michie, J. (1990) Cognitive models from subcognitive skills, in *Knowledge-based Systems for Industrial Control*, McGhee, J., Grimble, M. and Mowforth, P. (eds), Stevenage: Peter Peregrinus.

Mingers, J. (1989a) An empirical comparison of pruning methods for decision-tree induction. *Machine Learning*, **4**, 227–243.

Mingers, J. (1989b) An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, **3**, 319–342.

Pagallo, G. and Haussler, D. (1990) Boolean feature discovery in empirical learning. *Machine Learning*, **5**, 71–99.

Press, S. (1989) *Bayesian Statistics*, Wiley, New York.

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

Quinlan, J. (1988) Simplifying decision trees, in *Knowledge Acquisition for Knowledge-Based Systems*, Gaines, B. and Boose, J. (eds), Academic Press, London, pp. 239–252.

Quinlan, J., Compton, P., Horn, K. and Lazarus, L. (1987) Inductive knowledge acquisitions: A case study, in *Applications of Expert Systems*, Quinlan, J. (ed.), Addison-Wesley, London.

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

Ripley, B. (1987) An introduction to statistical pattern recognition, in *Interactions in Artificial Intelligence and Statistical Methods*, Unicom, Gower Technical Press, Aldershot, pp. 176–187.

Rissanen, J. (1989) *Stochastic Complexity in Statistical Enquiry*, World Scientific, Section 7.2.

Rodriguez, C. (1990) Objective Bayesianism and geometry, in *Maximum Entropy and Bayesian Methods*, Fougère, P. (ed.), Kluwer, Dordrecht.

Stewart, L. (1987). Hierarchical Bayesian analysis using Monte Carlo integration: computing posterior distributions when there are many possible models. *The Statistician*, **36**, 211–219.

Utgoff, P. (1989). Incremental induction of decision trees. *Machine Learning*, **4**, 161–186.

Wallace, C. and Patrick, J. (1991). Coding decision trees. Technical Report 151, Monash University, Melbourne, submitted to *Machine Learning*.

Weiss, S., Galen, R. and Tadepalli, P. (1990) Maximizing the predictive value of production rules. *Artificial Intelligence*, **45**, 47–71.