# Experiences with OB1, An Optimal Bayes Decision Tree Learner

John G. Cleary and Leonard E. Trigg
Department of Computer Science
University of Waikato
Hamilton, New Zealand
{jcleary, trigg}@cs.waikato.ac.nz

**Abstract**

In machine learning, algorithms for inferring decision trees typically choose a single "best" tree to describe the training data, although recent research has shown that classification performance can be significantly improved by voting predictions of multiple, independently produced decision trees. This paper describes a new algorithm, OB1, that weights the predictions of any scheme capable of inferring probability distributions. We described an implementation, OB1, that includes *all* decision trees, as well as naive Bayesian models. Results indicate that OB1 is a very strong robust learner and makes plausible the claim that it successfully subsumes other techniques such as boosting and bagging that attempt to combine many models into a single prediction.

**Keywords**: Option trees, Bayesian Statistics, Decision trees.
**Email address of contact author**: jcleary@cs.waikato.ac.nz
**Phone number of contact author**: +64 7856 2889

## 1 Introduction

The standard approach for inferring decision trees from training data is to choose a single "best" tree to describe the training data [13]. Following Occam's Razor, the best tree is usually defined as the simplest tree fitting the data. This is usually expressed as the Maximum Likelihood Principle and more generally as the Minimum Description Length (MDL) principle [3]. These tree-based systems have performed well on a variety of real-world tasks, and have the advantage that decision trees are relatively easy to understand. This approach has some problems however; in particular what is the basis for choosing only the simplest tree. Also, it is easy to overfit and choose a tree that is large and complex but which predicts less well than simpler trees on new data.

An alternative approach is rooted in the philosophy of Epicurus [12], who believed *all* hypotheses fitting the data should be retained. Along these lines recent research has shown that classification performance can be significantly improved by voting predictions of multiple decision trees [14]. A similar approach weights different trees and forms a weighted sum of their predictions. For example, Buntine [4] forms option trees which combine some subset of the possible trees by weighted averaging at internal nodes of the tree. A more dramatic approach is the Context Tree Weighting (CTW) models [16] used in text compression. CTW sums over all possible models, weighted according to the probability of all the preceding data. Volf [15] has applied the CTW approach to machine learning to derive MDL decision trees.

It might seem that summing over *all* possible models would be very expensive. However, it is possible to use the internal structure of the models to avoid recomputing values repeatedly. The result is a collapse in both the space and time required to construct and use the data.

This paper describes an optimal Bayes learner called OB1 which has an Epicurean foundation. We show how the set of all decision trees can be represented compactly and efficiently. Some experiments are then described which compare OB1 against a number of other machine learning techniques. It is shown that depth bounded OB1 can perform significantly worse than a naive Bayesian [9] learner. This then motivates the inclusion of naive Bayesian models into OB1. Comparisons are then made between the OB1 system and a range of learning algorithms on a number of datasets taken from the UCI repository. The comparison is evaluated using both an information gain measure and classification accuracy of the schemes. The paper concludes with a summary of what has been achieved and of future developments in the implementation of OB1.

## 2 Constructing OB1 Option Trees

Superfised machine learning is concerned with the problem of predicting a set of data given instances of the data. Given some set of data values $D = \{v_1, \ldots, v_N\}$ and for each data value some set of corresponding instances $\{x_1, \ldots, x_N\}$. Each instance $x$ is a vector $\langle a_1, \ldots, a_m \rangle$ where the $a_i$ are the individual attribute values for the instance. The problem then is to predict the next data value $v$ given some attributes $x$ and a set of prior data $D$. Prediction can be done two ways. One is to make a categorical prediction by selecting one of the possible values for $v$. The other is to produce a probability distribution over the possible values of $v$, this is written $P(v|D)$, the conditional probability of $v$ given $D$. In what follows it is simpler to use the probability $P(D)$ that is, the probability of the entire set of data occurring. This can be interchanged with the conditional probability using the relation

$$P(v|D) = \frac{P(D \cup \{v\})}{P(D)} \tag{1}$$

In this paper we will restrict ourselves to the case where there are a finite number of discrete attributes and data values. In particular we do not consider the case of real values. The fundamental idea of Epicurean learning is that $P(D)$ can be computed by considering many different possible hypotheses. It is assumed that given an hypothesis we can compute a probability for the data. The fundamental relationship that is used to compute $P(D)$ is :

$$P(D) = \sum_{h \in H} P(D|h)P(h)$$

$P(D|h)$ is the probability of the data given the hypothesis $h$. $H$ is the set of all possible hypotheses that we are considering. $P(h)$ is the prior likelihood of the hypothesis. $-\log_2(P(h))$ can be considered the complexity of the hypothesis—that is, the number of bits necessary to specify it.

Given a set of possible hypotheses we want to associate a recursive AND-OR tree structure with them. Each node $I$ will have associated with it a set of hypotheses, $H_I$, a subset of the data, $D_I$, a prior $P_I(h)$ over $H_I$, and a probability estimate

$$P_I(D_I) = \sum_{h \in H_I} P_I(D_I|h) P_I(h)$$

The importance of the tree structure is that $P_I(D_I)$ can be expressed recursively in terms of its value on the child nodes. There are three types of nodes in the tree: AND, OR and LEAF nodes.

At an AND node $I$ the data $D_I$ is split according to the values of one of the attributes (this corresponds to an interior node in a standard decision tree). So for some attribute which can take on the values $a_1, \ldots, a_l$ there will be $l$ children $I_1, \ldots, I_l$. Thus $D_{I_i}$ is all the values in $D_I$ whose corresponding attribute has the value $a_i$. The set of hypotheses $H_I$ is all decision trees whose top level test is on the selected attribute. The set of hypotheses $H_{I_i}$ is the same for each child, that is, the set of all decision trees that never test the selected attribute (again). Each hypothesis $h$ in $H_I$ splits into subhypotheses $h_1, \ldots, h_l$, that is $h$ is composed of disjoint parts which apply to the disjoint data sets $D_{I_i}$. This implies that $P(D_I|h) = \prod_i P(D_{I_i}|h_i)$. Similarly the priors can be computed as $P_I(h) = \prod_i P_{I_i}(h_i)$. Any subhypothesis in the children can be combined with other subhypotheses in the other children to form a complete hypothesis. So, at least in the case where we are summing over all possible hypotheses, $H_I$ is the cross product of the hypotheses in the children. Given all these relationships the overall probability of the data can be re-expressed as:

$$P_I(D_I) = \prod_i P_{I_i}(D_{I_i}) \tag{2}$$

An OR node $I$ segregates the set of hypotheses by the first attribute tested in the hypothesis. So an OR node will have one child, $I_i$, for each attribute and possibly some leaf nodes. For each non-LEAF child $I_i$ the set of hypotheses $H_{I_i}$ will be all hypotheses in $H_I$ with an intial test on the corresponding attribute. Each LEAF child will generate a probability using the statistics from all the values in $D_I$. (Some options for doing this are explained below). An OR-node corresponds to an option node in the work of Buntine [4]. The data set for each of the children is unchanged from the parent, that is, $D_I$. The priors for the children remain unchanged except for the assumption that all priors sum to 1. This requires that the priors at each child be renormalized by the term $S_{I_i} \equiv \sum_{h \in H_{I_i}} P_I(h)$. The new priors are then $P_{I_i}(h) = P_I(h)/S_{I_i}$ . Note that $\sum_{h \in H_I} P_I(h) = 1$ implies $\sum_i S_{I_i} = 1$.

The probability of the data at $I$ can be re-expressed as the weighted sum:

$$P_I(D) = \sum_i S_{I_i} P_{I_i}(D) \tag{3}$$

AND-nodes form their composite probability by taking the product of the probabilities from their children and the OR-nodes by taking a weighted sum of their children.

The LEAF nodes should use the statistics of the all the data in $D$ and return from this an estimate of $P(D)$. One of the simplest such estimators is the Laplace estimator. This assumes that there are an infinite number of hypotheses each corresponding to a different probability between 0 and 1 for the next symbol. In the case where the values in $D$ are binary, one of $a$ or $b$, and $c_a$ and $c_b$ are the number times $a$ and $b$ respectively has occurred in $D$ then using the integral form of Eqn.2

$$P(D) = \int_0^1 p^{c_a}(1-p)^{c_b} = \frac{c_a! c_b!}{(c_a + c_b + 1)!}$$

In the general form of this equation

$$P(D) = \frac{\prod_v c_v!}{(c + n - 1)!} \tag{4}$$

where $v$ ranges over all possible data values, $c = \sum_v c_v$ and $n$ is the number of different data values. Other estimators such as the Dirichlet estimator are possible here.

Given an AND-OR tree the most common use is to compute a conditional probability using Eqn.1 A simple recursive function can do this, using the following relationships .At an AND-node $I$

$$P_I(v|D_I) \quad = \quad \frac{\prod_i P_{I_i}((D \cup \{v\})_{I_i})}{\prod_i P_{I_i}(D_{I_i})}$$

However, $(D \cup \{v\})_{I_i}$ will be equal to $D_{I_i}$ except for the one case where the attribute associated with $v$ is equal to $a_i$. That is, most of the terms in the product above cancel giving

$$P_I(v|D_I) \quad = \quad P_{I_i}(v|D_{I_i})$$

At an OR-node $I$

$$\begin{aligned} P_I(v|D_I) &= \frac{\sum_i S_{I_i} P_{I_i}(D_I \cup \{v\})}{\sum_i S_{I_i} P_{I_i}(D_I)} \text{ from Eqn.3} \\ &= \frac{\sum_i S_{I_i} P_{I_i}(D_I) P_{I_i}(v|D_I)}{\sum_i S_{I_i} P_{I_i}(D_I)} \text{ using Eqn.1} \end{aligned}$$

At a LEAF node $I$ using the Laplace estimator the conditional probability can be computed as follows

$$\begin{aligned} P_I(v|D_I) &= \frac{(c_v+1)! \prod_{w \neq v} c_w!}{(\sum_w c_w + n)!} / \frac{\prod_w c_w!}{(\sum_w c_w + n - 1)!} \\ &= \frac{c_v + 1}{(\sum_w c_w + n)} \end{aligned}$$

In order to execute such an algorithm effectively it is only necessary to record the counts $c_v$ on the leaf nodes and the weights $W_{I_i} \equiv S_{I_i} P_{I_i}(D_I)$ for each child $I_i$ of an OR-node $I$. Given such a set of weights the recursive computation of $P_I(v|D_I)$ of a node $I$ with children $I_i$ can be summarised as

$$P_I(v|D_I) = \begin{cases} P_{I_i}(v|D_{I_i}) & \text{where } I \text{ is an AND node and } I_i \text{ is selected} \\ \frac{\sum_i W_i P_{I_i}(v|D_I)}{\sum_i W_i} & \text{where } I \text{ is an OR node} \\ \frac{c_v+1}{(\sum_w c_w + n)} & \text{where } I \text{ is a LEAF node} \end{cases}$$

One difficulty with this approach is that the $W_i$ can be too small to store in a standard floating point representation which potentially could slow the execution of the algorithm. Fortunately, they can be renormalized by any factor and the equation for conditional probabilities will remain unchanged. In OB1 they are normalized so that $\sum_i W_i = 1$, and stored in a single precision floating point.

Typically OB1 is used in two phases. In the first phase training instances are supplied and the AND-OR tree is constructed together with the counts and weights. In the second phase test instances are supplied and the conditional probabilities are computed as above. The weights can be computed incrementally using an algorithm similar to that above. This relies on

$$P(D) = \prod_{i=1}^{N} P\left(v_i | \{v_1, \ldots v_{i-1}\}\right),$$

that is, the total probabilities can be computed incrementally as new data arrives. The procedure $W_I(v|D_I)$ below incrementally computes the weights and updates the counters on the leaves.

```
$W_I(v|D_I)$
    if $I$ is an AND node and $I_i$ is selected
        return $W_{I_i}(v|D_{I_i})$
    if $I$ is an OR node
        for all $I_i$ children of $I$ $W_i := W_i * W_{I_i}(v|D_I)$
        return $t := \sum_i W_i$
        for all $I_i$ children of $I$ $W_i := W_i/t$  {renormalise weights}
    if $I$ is a LEAF node
        $c_v := c_v + 1$
        $c := c + 1$
        return $c_v/c$
```

To initialise the counts the total counters $c$ are assigned $n$ the total number of possible values. The weights $W_i$ should be initialised to the renormalising factors $S_{I_i}$ There is complete freedom to choose the $S_{I_i}$ so long as they are positive and sum to one for the children of each OR-node . In fact they provide a very convenient way of specifying the priors. An approach that works well in practice is to set ?? what did we do?? each $S_{I_i} = 1/l$ where $l$ is the number of children of the OR-node.

Fig. ?? shows an example of an AND-OR tree. It is has two binary valued attributes. The figure shows the structure of the complete tree encompassing all possible decision trees for this case. As well the children of OR-nodes are labelled with their $S_I$ values. The figure also includes one example decision tree, the corresponding nodes in the full AND-OR tree are highlighted. It is notable that some of the subtrees are duplicates of each other. Fig. ?? shows a reduced form of the AND-OR tree which uses pointers to common subtrees to avoid any duplication. This reduces the size of the tree slightly in this case but for larger trees the savings can be dramatic. For example a full AND-OR tree with $a$ binary valued attributes will have approximately $2^a a!$ nodes while a form without duplicate subtrees will have (only) $3^a$. OB1 also uses another memory reduction technique by creating the tree lazily. Whenever a node has only one instance in its data set $D_I$ the subtree under it is not grown. Instead the node retains a reference to the single instance associated with it. Later, if another instance is inserted at $I$ the children will be created.

## 3 Initial evaluation

Initial evaluation of OB1 involved a comparison with a naive Bayes classifier [9] on a variety of datasets from the UCI repository [11] using the WEKA workbench [7]. The performance measure used for evaluation is a MDL significance measure, $S_f$, [5] which gives a more accurate measure of the amount of information captured by a scheme than simple percentage correct. In essence, the $S_f$ measure is the gain in bits when encoding the test instance class with respect to the scheme's predictions as opposed to encoding with respect to a naive method that produces predictions based only on the observed class frequencies during testing. Thus,

$$S_f = \sum_{i=1}^{N} -\log_2(P_{scheme}(i_c)) + \log_2(P_{naive}(i_c)),$$

where $N$ is the number of test instances, $i_c$ is the class of instance $i$, $P_{scheme}()$ is the scheme's predicted probability distribution and $P_{naive}()$ is the naive estimator, given by

$$P_{naive}(j) = \frac{T_j + 1}{T + C},$$

where $T_j$ is the number of times class $j$ has been seen so far in testing, $T$ is the number of instances tested so far, and $C$ is the number of classes.

Table 1 shows the $S_f$ in bits per instance ($S_f i$) averaged over 25 trials. For each trial, the dataset was randomly divided into two thirds training and one third testing. OB1 currently only handles nominal attributes, so all numeric attributes were discretised using Fayyad and Irani's [6] discretisation method on the training data to select cutpoints, which were then applied to the test data. OB1 was evaluated with tree depth bounded to 0, 1, 2, and 3 attributes (thus OB1 b0 predicts classes with the frequencies observed in the training data, irrespective of the predictor attributes). Trees deeper than 3 attributes are not considered in these experiments because for many of the datasets we had insufficient memory to generate deeper OB1 trees. Where naive Bayes results are significantly worse (according to a two-tailed, paired

| Dataset | NBayes | OB1 b0 | OB1 b1 | OB1 b2 | OB1 b3 |
|---|---|---|---|---|---|
| `anneal` | 1.03 | 0.06 $\oplus$ | 0.46 $\oplus$ | 0.95 $\oplus$ | 1.12 $\ominus$ |
| `audiology` | $-5.96$ | 0.09 $\ominus$ | 0.94 $\ominus$ | 1.73 $\ominus$ | 2.12 $\ominus$ |
| `autos` | $-0.72$ | 0.12 $\ominus$ | 0.53 $\ominus$ | 0.91 $\ominus$ | 1.06 $\ominus$ |
| `breast-cancer` | $-0.04$ | 0.03 $\ominus$ | 0.07 $\ominus$ | 0.07 $\ominus$ | 0.08 $\ominus$ |
| `credit-g` | 0.11 | 0.01 $\oplus$ | 0.09 | 0.10 | 0.11 |
| `diabetes` | 0.21 | 0.01 $\oplus$ | 0.17 $\oplus$ | 0.20 | 0.22 |
| `heart-c` | 0.43 | 0.17 $\oplus$ | 0.28 $\oplus$ | 0.34 $\oplus$ | 0.35 $\oplus$ |
| `heart-statlog` | 0.29 | 0.02 $\oplus$ | 0.16 $\oplus$ | 0.28 | 0.33 |
| `hypothyroid` | 0.38 | 0.01 $\oplus$ | 0.35 $\oplus$ | 0.40 $\ominus$ | 0.42 $\ominus$ |
| `iris` | 1.31 | 0.03 $\oplus$ | 1.28 | 1.34 | 1.34 |
| `labor` | 0.77 | 0.06 $\oplus$ | 0.37 $\oplus$ | 0.48 $\oplus$ | 0.49 $\oplus$ |
| `vote` | 0.15 | 0.01 $\oplus$ | 0.76 $\ominus$ | 0.78 $\ominus$ | 0.78 $\ominus$ |

Table 1: $S_f i$ for naive Bayes and OB1 on discretised UCI datasets

$t$-test at the 95% confidence level) than OB1, this is postfixed with $\ominus$, and where significantly better this is postfixed with $\oplus$.

The most obvious result seen in Table 1 is that the performance of OB1 improves with added tree depth. For some datasets, most of the class information can be captured in one or two attributes. For example, virtually all the information in `vote` is contained in one attribute, and for `iris` two attributes contain all the class information (although most of this can be obtained using only one attribute). Some datasets, such as `breast-cancer` and `credit-g` appear to contain very little class information. In general, we expect to see OB1 performance increase with tree depth up to a depth that captures the most domain information, then drop negligibly as tree depth continues to increase.

The poor performance of the naive Bayes classifier on `audiology` is noteworthy. Naive Bayes actually performs well on this dataset with respect to percentage correct, however the class probability it assigns is "all or nothing"—when it is incorrect, it often assigns a probability very close to zero to the actual class, and this accounts for its poor performance under the $S_f i$ measure. A more sophisticated naive Bayes implementation would be less certain of its prediction for these cases.

For most datasets however, naive Bayes performs very well, and on some datasets (such as `heart-c` and `labor`) it performs considerably better than the OB1 results shown (presumably because its attribute independence assumption isn't violated). The next section investigates whether we can make use of this to alleviate the need to build deep OB1 trees.

## 4 OB1 is Omnivorous

The only requirement for a hypothesis to be included in an OB1 tree is that the hypothesis be able to return a predicted probability for each class. Since naive Bayes models estimate the class probabilities, they are ideal for inclusion in an OB1 tree. At each OR-node in the OB1 tree, we can easily include a naive Bayes model that operates on all the attributes that have not been tested previously in the tree, since the attribute-value counts needed for a naive Bayes model can be obtained by visiting the LEAF nodes attached to each OR node below each attribute split (thus these "free" naive Bayes models cannot be placed at the bottom level of an OB1 tree). The only extra storage associated with each naive Bayes model is its hypothesis weight.

In theory, since OB1 now includes naive Bayes models as competing hypotheses, it should never perform more than a couple of bits worse than naive Bayes alone (except where OB1 trees are bounded to depth 0, since there is not enough information in the OB1 tree for a naive Bayes model to be included). However it turns out that this is not always the case. For example, the `anneal` dataset gives virtually identical results to those shown in Table 1, even though naive Bayes performs very well. To understand this, we can examine the OB1's model weights as training progresses. Figure 1 shows the weights of several models, when OB1 is bounded to consider at most single-attribute splits. *Default* is the order-0 model weight, *Naive Bayes* is the weight assigned to a naive Bayes model that considers all attributes, *'family'* is the weight assigned to the hypothesis that splits on the 'family' attribute, and *'surface quality'* is the weight assigned to the hypothesis that splits on the 'surface quality' attribute. Other hypotheses (that split on other attributes), are not shown.

Initially the hypothesis with the highest weight is *Default*. Between 50 and 230 training instances *'family'* and *'surface quality'* are often considered the best hypotheses, with *'surface quality'* winning out eventually. The weight assigned to *Naive Bayes* drops rapidly during the inital stages of training—so low that it cannot dominate the other hypotheses at the end of training, although its weight is consistently climbing. A further hundred or so training instances would probably see the naive Bayes model dominate

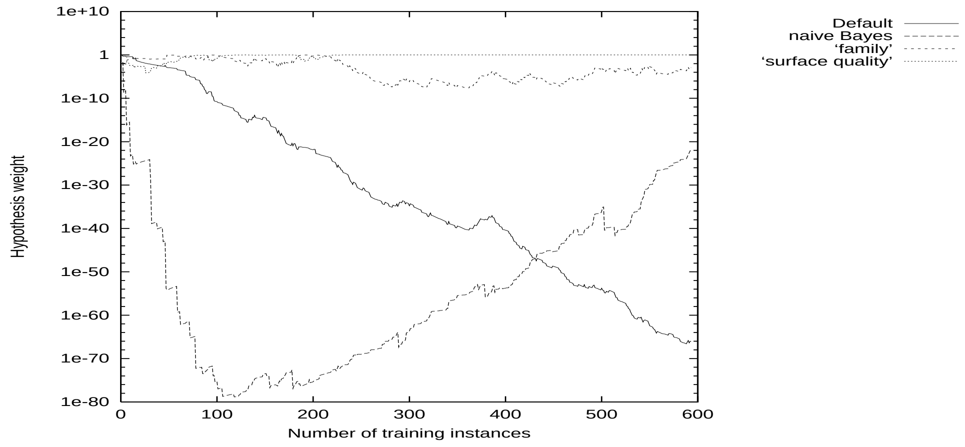

Figure 1: OB1 Hypothesis weight changes during training

the other hypotheses.

The current hypothesis weighting method effectively measures hypothesis performance throughout training, rather than the expected performance on future data. In this case, the naive Bayes models within OB1 are penalised for poor performance during the early stages of training, even though naive Bayes performs well during testing. The next section describes a new weighting method that addresses this problem.

## 5 Hold-One-Out Hypothesis Weighting

A common method of evaluating machine learning schemes is hold-one-out evaluation, where the scheme is trained on all instances but one, and tested on the remaining instance. This process is repeated until all instances have been tested. Intuitively, this should give the best possible measure of expected performance on new data, since the maximum data possible is used to train the scheme. However, hold-one-out evaluation can be grossly incorrect in some circumstances (for example, a dataset containing equal numbers of instances of each class will result in every instance being misclassified during hold-one-out evaluation). With more sophisticated performance measures that operate on predicted probabilities rather than the most likely class, this

| Dataset | NBayes | OB1 b0 | OB1 b1 | OB1 b2 | OB1 b3 |
|---|---|---|---|---|---|
| `anneal` | 1.03 | 0.06 $\oplus$ | 1.03 | 1.04 | 1.14 $\ominus$ |
| `audiology` | $-5.96$ | 0.09 $\ominus$ | 0.94 $\ominus$ | 1.72 $\ominus$ | 2.07 $\ominus$ |
| `autos` | $-0.72$ | 0.12 $\ominus$ | 0.52 $\ominus$ | 0.95 $\ominus$ | 1.02 $\ominus$ |
| `breast-cancer` | $-0.04$ | 0.03 $\ominus$ | 0.08 $\ominus$ | 0.03 $\ominus$ | $-0.01$ |
| `credit-g` | 0.11 | 0.01 $\oplus$ | 0.10 | 0.11 | 0.10 |
| `diabetes` | 0.21 | 0.01 $\oplus$ | 0.21 | 0.23 $\ominus$ | 0.23 $\ominus$ |
| `heart-c` | 0.43 | 0.17 $\oplus$ | 0.43 | 0.50 $\ominus$ | 0.49 $\ominus$ |
| `heart-statlog` | 0.29 | 0.02 $\oplus$ | 0.29 | 0.34 $\ominus$ | 0.34 $\ominus$ |
| `hypothyroid` | 0.38 | 0.01 $\oplus$ | 0.38 | 0.41 $\ominus$ | 0.42 $\ominus$ |
| `iris` | 1.31 | 0.03 $\oplus$ | 1.31 | 1.34 $\ominus$ | 1.34 $\ominus$ |
| `labor` | 0.77 | 0.06 $\oplus$ | 0.76 | 0.73 | 0.73 |
| `vote` | 0.15 | 0.01 $\oplus$ | 0.76 $\ominus$ | 0.76 $\ominus$ | 0.77 $\ominus$ |

Table 2: $S_f i$ results for naive Bayes and OB1 incorporating naive Bayes models using hold-one-out weighting on discretised UCI datasets

is not a concern (in fact this is a good argument why more sophisticated measures than percent correct and error rate should be employed).

Hold-one-out evaluation is costly in terms of time, since the scheme is trained $N$ times (where $N$ is the number of instances). In the case of OB1 and naive Bayes, it is possible to compute the hold-one-out evaluation in a single pass. For naive Bayes, each test instance can be effectively removed from the model by decrementing the appropriate attribute value counts by one. For OB1 the situation is a little more complex, as the accumulated hypothesis weights also have to be adjusted to temporarily remove the influence of the current test instance.

The hold-one-out procedure provides a good estimate of the expected performance of a model, and this can be used to replace the initial OB1 weighting method. Rather than the hypothesis weights representing the product of incremental predicted probabilities, the new weighting method forms the hypothesis weights from the product of the hold-one-out probabilities. For an OB1 tree without naive Bayes models, the new weighting method can be computed incrementally as a direct replacement for the original method. However, the product of hold-one-out probabilities cannot be computed incrementally for naive Bayes models (and so the hold-one-out weighting cannot be computed incrementally for OB1 trees that incorporate naive Bayes models). We have therefore opted to build the OB1 tree in one pass to collect counts, and compute the hypothesis weights in a second pass.

In practise the total learning time is less than doubled, since the second pass does not require memory allocation.

Using the notation in Eqn.4 the probability of a data value $v$ if one instance of it is held out is $P(v|D - \{v\}) = \frac{c_v}{c+n-2}$. Each symbol $v$ is held out $c_v times$ so the total contribution of that symbol to the probability is $\left(\frac{c_v}{c+n-2}\right)^{c_v}$. Thus

$$P(D) = \prod_v \left(\frac{c_v}{c+n-2}\right)^{c_v} = (c+n-2)^c \prod_v c_v^{c_v}$$

Table 2 shows the $S_f i$ for OB1 when including naive Bayes models in the tree and using hold-one-out weighting. Experimental conditions are the same as for the previous experiments. The results are much improved over those in Table 1—now OB1 always performs at least within a small constant of the better of naive Bayes and the original OB1. OB1 now performs considerably better than both methods alone on some datasets (such as `heart-c`, where the dominant hypotheses contain naive Bayes models at the lower levels of the tree.

## 6 Selecting a Single Model Decreases Performance

For the purposes of extracting a model that can be easily understood by humans, we can select the frontier of the OB1 tree with the highest weight. We can also produce predictions based only on the chosen frontier. Table 3 shows a comparison of the performance of the dominant frontier (pruned) with the performance of all frontiers (unpruned), for each of the depth bounds. All runs incorporated hold-one-out weighting, and included naive Bayes models among hypotheses. Results marked with $\oplus$ indicate where the unpruned model performed significantly better than the pruned model, using a two-tailed, paired $t$-test at the 95% confidence level. In none of the cases did the pruned model perform significantly better than the unpruned model.

To confirm that selecting a single model can significantly degrade performance, we performed another comparison, this time with the optimal depth 1 and depth 2 hypotheses, as determined by T1 and T2 respectively [2]. Since T1 and T2 do not produce probability distributions for each prediction, we

| Dataset | b1 | b1 Pr | b2 | b2 Pr | b3 | b3 Pr |
|---|---|---|---|---|---|---|
| `anneal` | 1.03 | 1.03 | 1.04 | 1.02 $\oplus$ | 1.14 | 1.07 $\oplus$ |
| `audiology` | 0.94 | 0.91 $\oplus$ | 1.72 | 1.68 | 2.07 | 1.97 $\oplus$ |
| `autos` | 0.52 | 0.49 $\oplus$ | 0.95 | 0.85 $\oplus$ | 1.02 | 0.51 $\oplus$ |
| `breast-cancer` | 0.08 | 0.07 | 0.03 | $-0.04$ $\oplus$ | $-0.01$ | $-0.15$ $\oplus$ |
| `credit-g` | 0.10 | 0.10 | 0.11 | 0.09 $\oplus$ | 0.10 | 0.09 $\oplus$ |
| `diabetes` | 0.21 | 0.21 | 0.23 | 0.22 | 0.23 | 0.22 $\oplus$ |
| `heart-c` | 0.43 | 0.43 | 0.50 | 0.48 $\oplus$ | 0.49 | 0.46 $\oplus$ |
| `heart-statlog` | 0.29 | 0.29 | 0.34 | 0.31 $\oplus$ | 0.34 | 0.28 $\oplus$ |
| `hypothyroid` | 0.38 | 0.38 | 0.41 | 0.40 $\oplus$ | 0.42 | 0.41 $\oplus$ |
| `iris` | 1.31 | 1.31 | 1.34 | 1.32 $\oplus$ | 1.34 | 1.32 $\oplus$ |
| `labor` | 0.76 | 0.76 | 0.73 | 0.63 $\oplus$ | 0.73 | 0.63 $\oplus$ |
| `vote` | 0.76 | 0.76 | 0.76 | 0.70 $\oplus$ | 0.77 | 0.68 $\oplus$ |

Table 3: $S_f i$ for OB1 (with hold-one-out weighting) unpruned vs pruned (Pr) results

| Dataset | OB1 b1 | T1 | OB1 b2 | T2 |
|---|---|---|---|---|
| `anneal` | 78.31 | 83.40 $\ominus$ | 93.31 | 76.73 $\oplus$ |
| `audiology` | 44.83 | 21.30 $\oplus$ | 58.34 | 23.90 $\oplus$ |
| `autos` | 48.51 | 46.17 | 65.14 | 65.89 |
| `breast-cancer` | 70.27 | 68.16 $\oplus$ | 70.10 | 67.05 $\oplus$ |
| `credit-g` | 69.11 | 70.84 $\ominus$ | 71.84 | 69.44 $\oplus$ |
| `diabetes` | 74.02 | 74.02 | 74.79 | 73.95 |
| `heart-c` | 71.81 | 72.19 | 73.48 | 72.12 $\oplus$ |
| `heart-statlog` | 71.30 | 71.30 | 72.52 | 70.57 |
| `hypothyroid` | 96.61 | 96.63 | 98.52 | 98.43 $\oplus$ |
| `iris` | 92.39 | 92.47 | 93.96 | 92.24 $\oplus$ |
| `labor` | 79.37 | 74.32 $\oplus$ | 85.47 | 83.79 |
| `vote` | 95.70 | 95.84 | 95.43 | 95.05 |

Table 4: Percentage correct for OB1 full trees (depth 1 and depth 2) and the best tree (obtained from T1 and T2), on discretised UCI datasets

| Dataset | OB1 | 1R | C4.8 | 1NN | 5NN | NBayes | T1 | T2 |
|---|---|---|---|---|---|---|---|---|
| `anneal` | 98.5 | 83.4 ⊕ | 98.4 | 98.0 ⊕ | 97.3 ⊕ | 95.9 ⊕ | 83.4 ⊕ | 76.7 ⊕ |
| `audiology` | 69.3 | 2.8 ⊕ | 77.0 ⊖ | 70.2 | 59.7 ⊕ | 58.0 ⊕ | 21.3 ⊕ | 23.9 ⊕ |
| `autos` | 70.2 | 46.1 ⊕ | 67.6 | 76.2 ⊖ | 59.7 ⊕ | 63.0 ⊕ | 46.2 ⊕ | 65.9 ⊕ |
| `breast-cancer` | 69.7 | 68.1 ⊕ | 69.9 | 69.1 | 73.2 ⊖ | 72.1 ⊖ | 68.2 ⊕ | 67.1 ⊕ |
| `credit-g` | 73.5 | 70.8 ⊕ | 71.5 ⊕ | 68.7 ⊕ | 71.8 ⊕ | 74.0 ⊖ | 70.8 ⊕ | 69.4 ⊕ |
| `diabetes` | 75.8 | 74.0 ⊕ | 75.2 ⊕ | 68.9 ⊕ | 73.7 ⊕ | 76.2 | 74.0 ⊕ | 74.0 ⊕ |
| `heart-c` | 80.5 | 72.2 ⊕ | 75.3 ⊕ | 76.8 ⊕ | 80.7 | 81.9 ⊖ | 72.2 ⊕ | 72.1 ⊕ |
| `heart-statlog` | 80.2 | 71.3 ⊕ | 78.7 | 76.8 ⊕ | 80.0 | 81.4 ⊖ | 71.3 ⊕ | 70.6 ⊕ |
| `hypothyroid` | 99.0 | 92.3 ⊕ | 88.9 ⊕ | 97.6 ⊕ | 96.9 ⊕ | 97.9 ⊕ | 96.6 ⊕ | 98.4 ⊕ |
| `iris` | 93.2 | 92.5 | 93.3 | 93.7 | 93.0 | 92.8 | 92.5 | 92.2 |
| `labor` | 92.4 | 73.9 ⊕ | 67.0 ⊕ | 84.6 ⊕ | 86.7 ⊕ | 92.6 | 74.3 ⊕ | 83.8 ⊕ |
| `vote` | 95.0 | 93.8 ⊕ | 92.5 ⊕ | 92.7 ⊕ | 93.2 ⊕ | 90.5 ⊕ | 95.8 ⊖ | 95.1 |

Table 5: Percentage correct for machine learning schemes, on discretised UCI datasets, relative to OB1 using hold-one-out weighting, naive Bayes models, with tree depth bound to 3 attributes

use percentage correct rather than $S_f i$ for evaluation. The results are shown in Table 4. Where OB1 b1 performs significantly better than T1 this is postfixed with ⊕, and where OB1 b1 performs significantly worse than T1 this is postfixed with ⊖ (similarly for T2 with OB1 b2).

There are few significant differences between T1 and OB1 b1—T1 performs better than OB1 b1 for two datasets, and worse for three. In contrast, T2 often performs significantly worse than OB1 b2 (7 out of the 12 datasets), and never significantly better.

## 7 Comparison With Other Machine Learning Methods

Schemes employed in this comparison were 1R [8], C4.8 [13], IB1 with 1 and 5 nearest neighbours [1], T1 and T2 [2], and a naive Bayes classifier [9], using default settings. OB1 settings are currently considered default: incorporating hold-one-out weighting, including naive Bayes models, and tree depth bound to three attributes.

Table 5 shows the percent correct results. Where a scheme performs significantly better than OB1 this is postfixed with ⊖, and where a scheme performs significantly worse than OB1 this is postfixed with ⊕. The results in Table 5 are impressive, with 8 cases where a scheme performs significantly better than OB1, and 59 cases where schemes perform significantly worse than OB1. As OB1 is still a work in progress, we intend to perform more detailed comparisons with other schemes, including boosted algorithms.

## 8 Conclusions and Future Work

In this paper we have presented a new algorithm, OB1, which combines many possible decision trees by weighting their predictions. The basic algorithm is capable of incorporating any scheme capable of inferring probability distributions. We described an implementation that includes decision trees, as well as naive Bayesian models.

The algorithm permits a single best tree with the highest weight to be selected from all the trees. However, experiments show that using this single tree never performs significantly better than the weighted sum and in some cases it performs much worse. Also when incorporating naive Bayesian models there are cases where OB1 performs better than either naive Bayes on its own or OB1 without naive Bayes models. This indicates that the weighted sum approach is effectively dealing with the problems of combining many models while at the same time avoiding overfitting.

OB1 was compared agaisnt a range of other techniques including both decision tree learners and instance based techniques. Using an information gain metric OB1 never performs significantly worse than any other technique on any of the tested data sets. Using accuracy as a comparison metric it was significantly worse than a naive Bayes in some cases and significantly better in others. Otherwise there were very few cases where OB1 was significantly worse. Overall these results indicate that OB1 is a very strong robust learner and makes plausible the claim that it successfully subsumes other technqiues such as boosting and bagging that attempt to combine many models into a single prediction.

We are currently pressing ahead with extending the range of OB1. Currently it is unable to use real valued attributes without prior discretisation. In theory there is no problem with real valued attributes, one just sums over all possible discretisations. Unfortunately, this causes too big an explosion in memory usage even for us. So first we are going to deal more intelligently with limiting memory than the current crude depth bounds. Initially we intend to test a simple heuristic based on the pruning the lowest weight models when memory is exhausted. The ability to deal with real valued attributes will significantly extend the range of data sets on which we can realisticly test OB1 and we intend to examine the large agricutural data sets available from the WEKA project [7].

Another more difficult problem is dealing effectively with real valued class information. We would like to do this while remaining within the complexity theoretic approach developed here. It is unclear how this can be done effectively.

Another approach to restraining memory usage is to use prior meta-data to restrict the summing space. This includes such information as functional dependencies between attributes as well as independence relations between attributes, as in Bayesian networks.

## References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Peter Auer, Robert C. Holte, and Wolfgang Maass. Theory and applications of agnostic PAC-learning with small decision trees. In Prieditis A. and Russell S., editors, *Proceedings of the 12th International Conference on Machine Learning (ICML95)*, San Francisco, CA, 1995. Morgan Kaufmann.

[3] R.A. Baxter and J.J. Oliver. MDL and MML: Similarities and differences (introduction to minimum encoding inference—part III). Technical Report Technical Report 207, Department of Computer Science, Monash University, Australia, 1994.

[4] Wray Buntine. Classifiers: A theoretical and empirical study. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991. Morgan Kaufmann.

[5] J. Cleary, S. Legg, and I. H. Witten. An MDL estimate of the significance of rules. In *Proceedings of the Information, Statistics and Induction in Science Conference*, pages 43–53, Melbourne, Australia, 1996.

[6] Usama M. Fayyad and Keki B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[7] G. Holmes, A. Donkin, and I. H. Witten. WEKA: A machine learning workbench. In *Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems*, Brisbane, Australia, 1994. [webpage at http://www.cs.waikato.ac.nz/∼ml/].

[8] Robert Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[9] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228, San Jose, CA, 1992. AAAI Press.

[10] M. Li and P. Vitanyi. *Introduction to Kolmogorov Complexity and its applications.* Springer-Verlag, New York, 1993.

[11] Christopher J. Merz and Patrick M. Murphy. *UCI Repository of Machine Learning Data-Bases.* University of California, Department of Information and Computer Science, Irvine, CA, 1996.

[12] W.J. Oates. *The Stoic and Epicurean Philosophers: The Complete Extant Writings of Epicurus, Epictetus, Lucretius, Marcus Aurelius.* Random House, New York, 1957.

[13] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1994.

[14] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the 14th International Conference on Machine Learning (ICML97)*, pages 322–330. Morgan Kaufmann, 1997.

[15] P.A.J Volf. Deriving MDL-decision trees using the context maximizing algorithm. Master's thesis, Department of Electrical Engineering, Eindhoven University of Technology, 1994.

[16] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, May 1995.