Applying the Weak Learning Framework to Understand and Improve C4.5

Tom Dietterich
Oregon State University
tgd@research.cs.orst.edu

Michael Kearns
AT&T Research
mkearns@research.att.com

Yishay Mansour
Tel Aviv University
mansour@math.tau.ac.il

1 Introduction

There has long been a chasm between theoretical models of machine learning and practical machine learning algorithms. For instance, empirically successful algorithms such as **C4.5** and backpropagation have not met the criteria of the PAC model and its variants. Conversely, the algorithms suggested by computational learning theory are usually too limited in various ways to find wide application. The theoretical status of decision tree learning algorithms is a case in point: while it has been proven that **C4.5** (and all reasonable variants of it) fails to meet the PAC model criteria [2], other recently proposed decision tree algorithms that do have non-trivial performance guarantees unfortunately require membership queries [6, 13].

Two recent developments have narrowed this gap between theory and practice—not for the PAC model, but for the related model known as *weak learning* or *boosting*. First, an algorithm called **Adaboost** was proposed that meets the formal criteria of the boosting model and is also competitive in practice [10]. Second, the basic algorithms underlying the popular **C4.5** and **CART** programs have also very recently been shown to meet the formal criteria of the boosting model [12]. Thus, it seems plausible that the weak learning framework may provide a setting for interaction between formal analysis and machine learning practice that is lacking in other theoretical models.

Our aim in this paper is to push this interaction further in light of these recent developments. In particular, we perform experiments suggested by the formal results for **Adaboost** and **C4.5** within the weak learning framework. We concentrate on two particularly intriguing issues.

First, the theoretical boosting results for top-down decision tree algorithms such as **C4.5** [12] suggest that a new splitting criterion may result in trees that are smaller and more accurate than those obtained using the usual information gain. We confirm this suggestion experimentally.

Second, a superficial interpretation of the theoretical results suggests that **Adaboost** should vastly outperform **C4.5**. This is not the case in practice, and we argue through experimental results that the theory must be understood in terms of a measure of a boosting algorithm's behavior called its *advantage sequence*. We compare the advantage sequences for **C4.5** and **Adaboost** in a number of experiments. We find that these sequences have qualitatively different behavior that explains in large part the discrepancies between empirical performance and the theoretical results. Briefly, we find that although **C4.5** and **Adaboost** are both boosting algorithms, **Adaboost** creates successively "harder" filtered distributions, while **C4.5** creates successively "easier" ones, in a sense that will be made precise.

2 C4.5 and Adaboost

In this section, we describe the two learning algorithms we will examine. Both algorithms take a finite *training sample* $S = \{\langle x_i, f(x_i)\rangle\}_{i=1}^{m}$ of m labeled examples as input. The x_i are points in some *instance space* X, and f is the boolean *target function* over X. The goal of both algorithms is to find a function with small training error on S in as few "rounds" as possible. The notion of a round will become clear shortly, but it can be thought of as a single step in which an algorithm increases the complexity of its hypothesis, such as splitting a leaf in the growing phase of **C4.5**.

2.1 A Top-Down Decision Tree Algorithm

It will be helpful to think of both algorithms as beginning with the *empirical distribution* P_S on S that gives weight $1/m$ to each of the m instances x_i in $S = \{\langle x_i, f(x_i)\rangle\}_{i=1}^{m}$; small training error on S is then equivalent to small error with respect to the distribution P_S. For the decision tree algorithm, we can now define some important quantities. If T is any decision tree, we let $leaves(T)$ denote its leaves. For each $\ell \in leaves(T)$, we define $w(\ell)$ to be the probability with respect to P_S that a random x reaches ℓ, and $q(\ell)$ to be the probability that $f(x) = 1$ *given* that x reaches

TopDown$_F(S, N)$:

1 Initialize T to be the single-leaf tree, with binary label equal to the majority class of the sample S.
2 **for** $t = 1$ to N:
3 $\quad \Delta_{best} \leftarrow 0$.
4 $\quad$ **for** each pair $(\ell, h) \in leaves(T) \times F$:
5 $\quad\quad \Delta \leftarrow H(T) - H(T(\ell, h))$.
6 $\quad\quad$ **if** $\Delta \geq \Delta_{best}$ **then** :
7 $\quad\quad\quad \Delta_{best} \leftarrow \Delta$; $\ell_t \leftarrow \ell$; $h_t \leftarrow h$.
8 $\quad T \leftarrow T(\ell_t, h_t)$.
9 Output T.

Figure 1: Algorithm **TopDown**$_F(S, N)$.

Adaboost$_F(S, N)$:

1 Initialize vector w to be $w_i = 1/m$, $1 \leq i \leq m$.
2 **for** $t = 1$ to N:
3 $\quad$ Let the distribution P_t be w normalized, so $P_t[x_i] = w_i / \sum_{j=1}^{m} w_j$.
4 $\quad$ Let $h_t \in F$ have the smallest error ϵ_t (with respect to f and P_t).
5 $\quad$ Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
6 $\quad$ Update w: if $h_t(x_i) = f(x_i)$, then $w_i = \beta_t \cdot w_i$, else w_i remains unchanged.
7 Output h such that $h(x) = 1$ if and only if $\sum_{t=1}^{N} \log(1/\beta_t) h_t(x) \geq (1/2) \sum_{t=1}^{N} \log(1/\beta_t)$.

Figure 2: Algorithm **Adaboost**$_F(S, N)$.

ℓ. We will call $w(\ell)$ the *weight* of the leaf. If $q(\ell) = 0$ or $q(\ell) = 1$ we say the leaf is *pure*. Notice that given T and S, $w(\ell)$ and $q(\ell)$ are easily computed. Let us assume that each leaf ℓ in T is labeled 0 if $q(\ell) < 1/2$, and is labeled 1 otherwise; we refer to this as the *majority* labeling. Under the majority labeling of the leaves of T, an expression for the training error $\hat{\epsilon}(T)$ of T on S is simply $\hat{\epsilon}(T) = \sum_{\ell \in leaves(T)} w(\ell) \min(q(\ell), 1 - q(\ell))$. Now if $H(q)$ is the binary entropy function, let us define the entropy of T:

$$H(T) = \sum_{\ell \in leaves(T)} w(\ell) H(q(\ell)). \qquad (1)$$

Since $H(q) \geq \min(q, 1 - q)$ for all q, $H(T) \geq \hat{\epsilon}(T)$. Thus if we can find a tree with small entropy, we have found a tree with small training error.

We need notation to describe incremental changes to the tree T. If $\ell \in leaves(T)$ and h is a boolean function over the input space X, we use $T(\ell, h)$ to denote the tree that is the same as T, except that now we "split" the leaf ℓ: we make a new internal node at ℓ and label this node by the function h. The newly created child leaves ℓ_0 and ℓ_1 (corresponding to the outcomes $h(x) = 0$ and $h(x) = 1$ at the new internal node) are labeled by their majority labels.

Our first algorithm, which we call **TopDown**$_F$, is given in Figure 1. It is parameterized by F, the class of *splitting functions* that can label the nodes of the tree. The algorithm takes as inputs the training sample S and the number of rounds N. At each round (i.e., each iteration of the outer **for** loop), a search is performed (inner **for** loop) for the leaf, and for the function in F labeling that leaf, that together maximize the resulting drop in entropy to the tree (information gain).

Notice that if N is sufficiently large, **TopDown**$_F$ will eventually split all impure leaves and grow the same *unpruned* tree as **C4.5**. Regardless of the value of N, the tree grown by **TopDown**$_F$ is always a subtree of the unpruned tree of **C4.5**. Thus, the main difference between **TopDown**$_F$ and **C4.5** is in their approaches to limiting the complexity of the resulting tree. In **C4.5**, the unpruned tree is grown until there are no impure leaves. Then, a pruning process is applied. Since the unpruned tree is always grown until purity in **C4.5**, the *order* in which leaves are split is irrelevant, and depth-first growth is acceptable. This means **C4.5** may choose to first split a leaf that reduces the entropy of the tree much *less* than some *later* split. In contrast, in **TopDown**$_F$ the complexity of the tree is limited by explicitly bounding the number of rounds N. In this case, since every split may be our last, the sensible thing to do is to always split next the leaf that maximizes the entropy reduction. **TopDown**$_F$ will choose the same function to label this split as **C4.5**, but may choose to perform the splits in a different order. Although there is experimental evidence that growth to purity followed by pruning may result in better performance than explicitly limiting the growth, algorithm **TopDown**$_F$ is easier to analyze. We believe that the theoretical results we will discuss shortly for **TopDown**$_F$ are relevant to **C4.5**, but are perhaps on the pessimistic side.

Note that we have left the choice of the splitting function class F as a parameter of the algorithm. While allowing a more expressive class F may reduce the entropy (and thus the training error) more rapidly, there are two costs for this. First, the search for the best function in F in the inner **for** loop of **TopDown**$_F$ becomes more expensive. Second, the relationship between the training and generalization errors will degrade as we increase the expressive power of F. The choice of F is a design decision that must be made with these trade-offs in mind. We shall assume that F is a rather simple class that can be searched rapidly, such as the individual attributes of the instance space.

2.2 A Boosting Algorithm

We are now ready to describe the second algorithm, whose code is given in Figure 2. We will not go into the detailed motivation for this algorithm, instead referring the reader to the paper of Freund and Schapire [10]. For our purposes, the most important idea is the creation of a *filtered distribution* at each

round. At round t, the algorithm "adds" to its final hypothesis the function h_t from F that has the lowest error on the filtered distribution P_t. The weight vector w (which determines the next filtered distribution through normalization) is then updated to *decrease* the relative weight of sample points on which h_t agrees with the target function, and to *increase* the relative weight of the sample points on which h_t errs. The next filtered distribution will thus be more concentrated on points that the preceding h_t's found "difficult". A reasonable intuition is that successive filtered distributions are focusing more and more on the hard part of the learning problem.

2.3 The Weak Hypothesis Assumption

Let us point out a number of similarities between the algorithms $\mathbf{TopDown}_F$ and $\mathbf{Adaboost}_F$. First, both algorithms proceed in rounds, where at each round a new function is chosen from the class F. These functions are incrementally *combined* to form a final hypothesis, although in rather different ways. In $\mathbf{TopDown}_F$, the function chosen at each round labels the internal node of a decision tree, while for $\mathbf{Adaboost}_F$, it becomes a member of a thresholded linear combination. For training error, we would expect both algorithms to perform better with more powerful classes F, but both would pay the computational and generalization error costs mentioned above.

Finally, both algorithms have some measure by which they choose the "best" function to add in each round. For $\mathbf{TopDown}_F$, this measure is the information gain resulting from the chosen split (Figure 1, line 5). For $\mathbf{Adaboost}_F$, it is the error on the filtered distribution for that round (Figure 2, line 4).

We will shortly discuss theoretical results for $\mathbf{Adaboost}_F$ that relate the training error after N rounds to the best errors ϵ_t (see Figure 2, line 4) on the sequence of filtered distributions $P_1, \ldots, P_N$; briefly, if these errors are just slightly smaller than the trivial value of $1/2$ (achieved by random guessing), very strong performance guarantees can be given for $\mathbf{Adaboost}_F$. But first, let us show that there is also a natural definition for the "filtered distributions" generated by $\mathbf{TopDown}_F$. If ℓ is any leaf of the current decision tree T, we can define P_ℓ to be the empirical distribution on only those instances in the sample S that reach ℓ. Now $q(\ell)$ is simply the probability of drawing a positive example from P_ℓ. The information gain of splitting T at ℓ is simply the information gain with respect to P_ℓ, weighted by $w(\ell)$.

What happens if, as for $\mathbf{Adaboost}_F$, we assume that on the sequence of distributions $P_{\ell_1}, \ldots, P_{\ell_N}$ (where ℓ_t is the t-th node split in the call to $\mathbf{TopDown}_F(S, N)$), there is always a function in F whose predictive power is slightly better than random guessing? The answer is that nothing happens, but for a superficial and easily remedied reason. Suppose that $q(\ell) = q$ is close to 1 (that is, the examples reaching ℓ are mainly positive). Then a function h that is always positive will have error $1 - q$, which is much better than random guessing, but will induce a trivial and useless split at ℓ. It turns out [12] that we need to slightly alter our definition of the filtered distributions for $\mathbf{TopDown}_F$. Let the *balanced* distribution P'_ℓ at ℓ be defined by $P'_\ell(x) = P_\ell(x)/(2q)$ if $f(x) = 1$ and $P'_\ell(x) = P_\ell(x)/(2(1-q))$ if $f(x) = 0$. Thus P'_ℓ is P_ℓ modified to give equal weight to the positive and negative examples of f. We redefine the t-th filtered distribution P_t for $\mathbf{TopDown}_F$ to be the distribution P'_{ℓ_t}, where ℓ_t is the leaf split on the t-th round; P_0 is the empirical distribution P_S on the training data.

Kearns and Mansour [12] show that if we assume there is always a function in F outperforming random guessing on the filtered distributions P'_{ℓ_t}, then there is a function in F giving a nontrivial information gain. This allows us to directly compare $\mathbf{Adaboost}_F$ and $\mathbf{TopDown}_F$ in the model sometimes known as *boosting* or *weak learning*, which we now define.

Definition 1 *For any $\gamma \in (0, 1/2]$, we say that F γ-satisfies the* **Weak Hypothesis Assumption** *(or* **WHA** *for short) with respect to f if for any distribution P over X, there is an $h \in F$ satisfying $\Pr_P[h(x) \neq f(x)] \leq 1/2 - \gamma$. We call γ the* **advantage** *(over random guessing).*

It is known that if F γ-satisfies the WHA with respect to f, then f can be well-approximated by thresholded linear combinations of functions from F in a sense that can be made precise [9]. Thus the WHA is not unlike the PAC model and other standard theoretical models, where one obtains leverage by making a priori assumptions on the form of the target function f. However, in the WHA this assumption is indirect, as we simply assume that the "simple" class F contains weak approximations to f on any distribution.

We now state results for the two algorithms under the WHA. The first is for $\mathbf{Adaboost}_F$.

Theorem 2.1 *(Freund and Schapire [10]) Let F be any class of boolean functions, let $\gamma \in (0, 1/2]$, and let f be any target boolean function that γ-satisfies the WHA with respect to F. Let S be any training sample of f, and let h denote the hypothesis output by $\mathbf{Adaboost}_F(S, N)$. Then for any ϵ, the training error of h on S is less than ϵ provided that*

$$N \geq \frac{1}{2\gamma^2} \ln \frac{1}{\epsilon}. \tag{2}$$

Kearns and Mansour [12] show that $\mathbf{TopDown}_F$ can also be profitably analyzed under the WHA. Perhaps the most significant aspect of their result is that it provides a performance guarantee for a popular and experimentally successful heuristic in an independently motivated theoretical model of learning.

Theorem 2.2 *(Kearns and Mansour [12]) Let F be any class of boolean functions, let $\gamma \in (0, 1/2]$, and let f be any target boolean function that γ-satisfies the WHA with respect to F. Let S be any training sample of f, and let T denote the tree output by* $\mathbf{TopDown}_F(S, N)$*. Then for any ϵ, the training error of T on S is less than ϵ provided that*

$$N \geq \left(\frac{1}{\epsilon}\right)^{c \log(1/\epsilon)/\gamma^2} \tag{3}$$

for some constant $c > 0$.

Two important remarks on these theoretical results are in order here. First, notice that in both theorems, the number of rounds required to achieve a desired training error is independent of the training sample size m. Thus, if m is sufficiently large compared to the given bounds, statements about generalization error can be obtained by standard Occam's Razor [4] or VC dimension arguments [3]. These statements will of course be different for the two algorithms due to the differing bounds on N and their differing hypothesis spaces; see the papers for details [10, 12].

Second, the bound given for $\mathbf{TopDown}_F$ in Theorem 2.2 is not the best possible for a top-down decision tree algorithm. By modifying the information gain criterion used by $\mathbf{TopDown}_F$, Kearns and Mansour are able to show an improved bound of

$$N \geq \left(\frac{1}{\epsilon}\right)^{c/\gamma^2} \tag{4}$$

where $c > 0$ is a constant. They also show that this bound is close to the best possible for any top-down decision tree algorithm.

2.4 Issues Raised by the Theoretical Results

There are two intriguing issues raised by the theoretical results given above, and their experimental investigation is the primary contribution of this paper.

First, the Kearns and Mansour results suggest that top-down decision tree algorithms such as **C4.5** might be improved by a change in the splitting criterion. We investigate this suggestion in detail in Section 3.

Second, even if we examine the improved bound of Equation (4), the difference between this bound and that given for $\mathbf{Adaboost}_F$ in Theorem 2.1 suggests that $\mathbf{Adaboost}_F$ should vastly outperform **C4.5**. In Section 4, using experiments due to Freund and Schapire [8] as our starting point, we find that the two algorithms are in fact rather comparable. We show experimentally that the discrepancy between this finding and the theory lies (at least in part) in our interpretation of the WHA advantage γ. We show that the advantage γ (or rather, the *sequence* of advantages $\gamma_1, \ldots, \gamma_N$ obtained on the filtered distributions $P_1, \ldots, P_N$) must be regarded as an algorithm-dependent and distribution-dependent quantity, and thus the theoretical bounds given for the two algorithms are incomparable. In other words, even though $\mathbf{Adaboost}_F$ has a better bound than $\mathbf{TopDown}_F$ for the same fixed advantage γ, in practice the advantage sequence for **C4.5** is "better" than that for $\mathbf{Adaboost}_F$, resulting in the approximate parity of the two algorithms.

3 An Improved Splitting Criterion?

As we have already mentioned, the bound given by Theorem 2.2 is not the best that can be obtained for a top-down decision tree algorithm. In particular, define $G(q) = 2\sqrt{q(1-q)}$, and in analogy with Equation 1, define

$$G(T) = \sum_{\ell \in leaves(T)} w(\ell) G(q(\ell)). \tag{5}$$

Kearns and Mansour prove that if we replace the entropy measure $H(T)$ in $\mathbf{TopDown}_F$ with the new measure $G(T)$ (that is, we change line 5 in Figure 1 to read $\Delta \leftarrow G(T) - G(T(\ell, h))$, but leave all other aspects of the algorithm unchanged) then Theorem 2.2 holds with the improved bound given by Equation (4). Thus a simple change in our splitting criterion function from $H(q)$ to $G(q)$ yields a polynomial rather than superpolynomial dependence on $1/\epsilon$.

The reason $G(q)$ gives a better bound than $H(q)$ is that it has better concavity. Consider splitting a node whose proportion of positive training examples is q into two child nodes whose proportions of positive training examples are p and r. Suppose that a fraction τ of the training examples are sent to the r child, and the remaining $1 - \tau$ examples go to the p child. Note that these split parameters must obey the equality $q = (1-\tau)p + \tau r$. In this situation, the information gain can be written as

$$H(q) - (1-\tau)H(p) - \tau H(r). \tag{6}$$

A geometrical interpretation of the information gain is shown in Figure 3, which plots $H(q)$ as a function of q. The figure shows vertical lines at p and r. The length of the short line segment descending from $H(q)$ to the line connecting $H(p)$ to $H(r)$ is the information gain. The fact that the information gain is non-zero is due to the concavity of the H curve.

In contrast, consider Figure 4. This figure plots the observed training set error $\min(q, 1-q)$ as a function of q. We can see that if we used this function in place of H in $\mathbf{TopDown}_F$, there would be no "gain" unless $p < 1/2 < r$. This is because both child nodes would get the same majority class label, so making this split would not make the tree more accurate on the training data (which is what $\min(q, 1-q)$ measures). The only reason to make the split is that it "makes progress" so that subsequent splits will be able to separate the two classes. Hence, the purpose of using a concave splitting criterion such as H is to provide some measure of

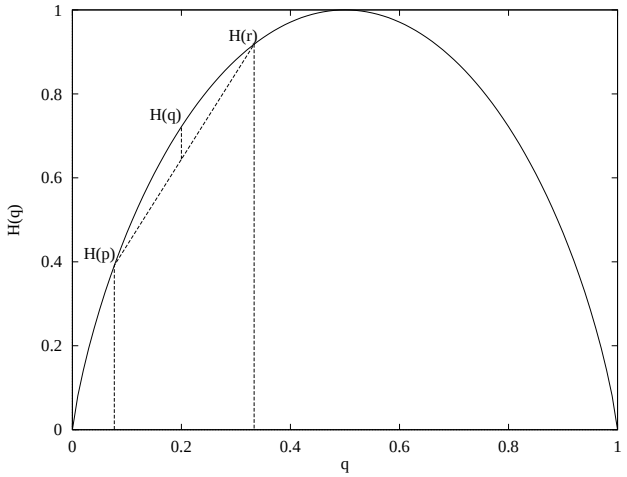

Figure 3: The entropy curve and the information gain.

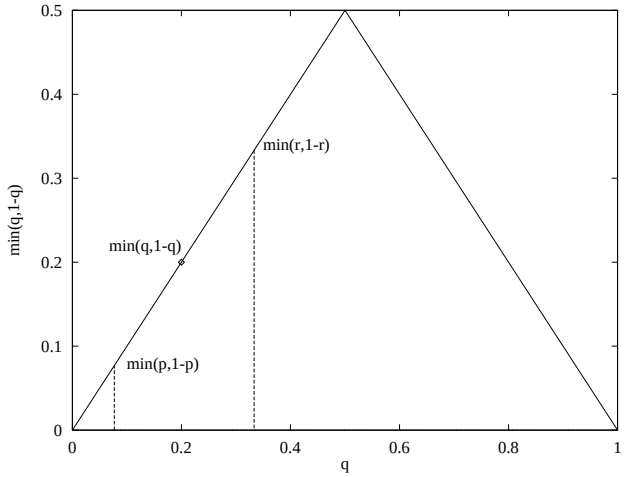

Figure 4: The $\min(q, 1-q)$ curve. No gain is obtained.

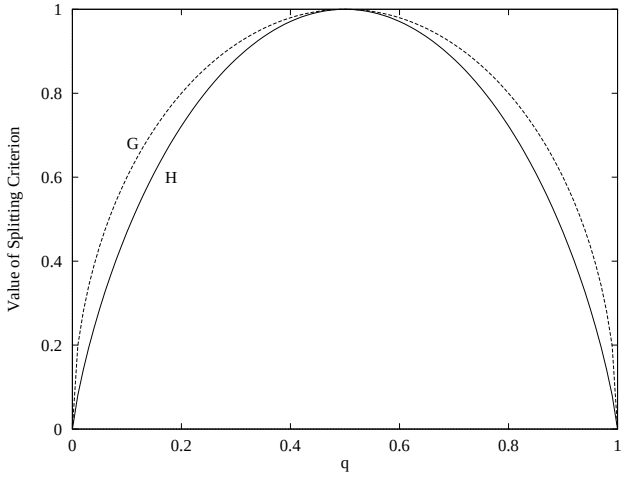

Figure 5: Comparison of G and H.

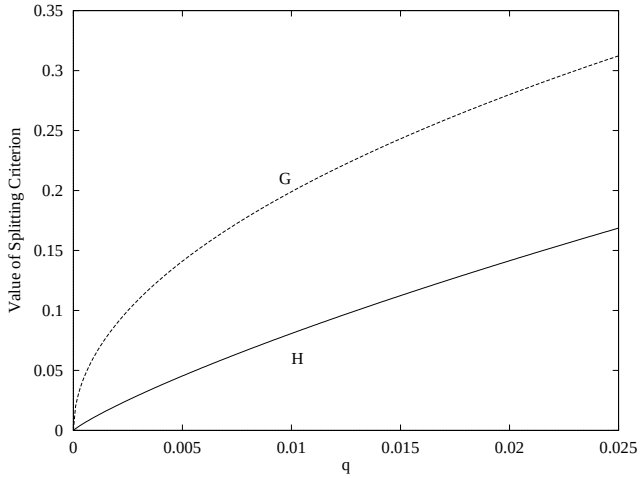

Figure 6: Comparison of concavity of G and H for small values of q.

"progress" during the tree-growing process even when p, q, and r are all on the same side of $1/2$.

Finally, Figure 5 plots the new splitting criterion $G(q)$ as a function of q and compares it to $H(q)$. We can see that G has improved concavity compared to H, especially for very small (or very large) values of q, as shown in Figure 6. This permits Kearns and Mansour to prove the tighter result given above.

3.1 Do G and H Differ in Practice?

Despite the improved bound for G and the geometric intuition, it is reasonable to ask whether G will actually outperform H in practice. Previous studies [14, 5] have generally found that decision tree algorithms are insensitive to the choice of the splitting criterion.

Let us be a little more precise about what Theorem 2.2 and Equation (4) say. Both results assert a relationship between tree size and error. Thus, for the same fixed tree size N, the theory predicts smaller error ϵ for the tree obtained using G compared to that obtained using H. Alternatively, if G and H result in trees with roughly the same error ϵ, the theory predicts that the tree obtained using G should be smaller than that obtained using H.

Table 1 shows the results of comparing G and H on nine challenging problems from the Irvine repository [15]. We modified **C4.5** Release 8 to implement the new G function. Our modified version could only handle 2-class problems, so we converted multiclass problems into problems of discriminating one class from the remaining classes. For this purpose, we chose the classes that were difficult to discriminate. In our comparisons, we used the **C4.5** parameters `-m 1 -g`, which force the tree to be grown all the way to pure leaves and which force **C4.5** to use the information gain rather than the gain ratio. We also disabled the

Table 1: Results of comparing **C4.5**(G) and **C4.5**(H). The first three columns show the % error on pruned trees and a 95% confidence interval on the differences. The remaining four columns show the sizes of the unpruned trees. The class chosen for discrimination against all other classes is shown in parentheses.

	% error	% error	95% confidence	Tree Size			
Problem	**C4.5**(G)	**C4.5**(H)	interval	**C4.5**(G)	**C4.5**(H)	t	Pr(t)
Letter Recognition ('H')	2.52	2.63	$[-0.02, 0.25]$	118.75	122.00	1.051	0.328
ISOLET ('E')	3.53	3.61	$[-0.31, 0.48]$	26.50	34.50	4.899	0.016**
ISOLET ('P')	3.26	3.62	$[-0.03, 0.80]$	29.00	37.50	4.977	0.016**
Annealing ('3')	3.78	3.94	$[-0.20, 0.61]$	23.50	32.50	2.286	0.106
Chess KR-vs-KP	1.69	1.83	$[-0.07, 0.37]$	49.75	52.50	2.906	0.062
Segmentation ('4')	3.21	4.06	$[0.38, 1.39]$**	21.00	22.00	1.000	0.139
Pima	28.91	30.66	$[-2.40, 5.87]$	98.00	95.00	3.000	0.205
Shuttle ('1')	0.04	0.08	$[0.02, 0.07]$**	20.00	25.00	2.402	0.096
Satellite ('4')	9.53	9.14	$[-1.36, -0.05]$**	104.50	112.00	1.202	0.316

new heuristic in Release 8 that penalizes continuous features having a large number of distinct values.

To test for significant differences, we employed the methodology advocated by Hinton et al. [11], in which the available training data are subdivided into n disjoint training sets and a single test set. Each algorithm is trained on each training set and the resulting hypotheses are all tested on the single test set. The results of classifying each test set example by each of the $2n$ hypotheses are tallied into a contingency table, and a 95% confidence interval is computed for the difference between the error rates of G and H using the ABC method for marginal homogeneity [7, 1]. This is a very sensitive test that controls for variations due to the different training sets.

In 8 of the 9 domains, the observed test error rate of **C4.5**(G) is lower than **C4.5**(H). The individual tests for significant differences find two domains where **C4.5**(G) is better and one where **C4.5**(H) is better. The combined results of all 9 domains, when analyzed by a sign test, show that **C4.5**(G) is performing better than **C4.5**(H) ($p < 0.039$). Based on these results, there is weak evidence that G produces better error rates than H, although the differences are not large. However, recall that the theory predicts that if G and H produce trees with similar error rates, the trees obtained using G should be smaller. Table 1 also shows the application of paired-differences t tests to compare the sizes of the unpruned decision trees. In all but one problem, the decision trees produced by G are smaller than those produced by H, although the difference is significant below the 0.05 level in only 2 problems. But if we combine all 38 trials in this table into a single t test, the difference is highly significant $p < 0.0002$. We conclude that G on the average produces smaller unpruned trees than H.

These experiments suggest that G may be a better splitting criterion than H, and that it merits further development (especially determining how it should be extended to handle more than two classes).

4 The Advantage Sequence

As pleased as we might be to give a nontrivial proof of performance for a **C4.5**-like algorithm in a standard theoretical learning model, and despite the small improvements that seem possible with the new splitting criterion discussed in Section 3, there is no avoiding the harsh reality that the bounds of Theorem 2.2 and Equation (4) are vastly worse than the bound of Theorem 2.1. Should we really believe that **C4.5** is considerably inferior to **Adaboost**$_F$?

As a partial answer to this question, consider Figure 7, which is due to Freund and Schapire [8]. This scatter plot compares the test errors of **Adaboost** [1] and **C4.5** (with pruning) on 27 problems from the UCI data repository. While **Adaboost** enjoys significant improvements over **C4.5** on many problems, there are a couple on which it performs considerably worse than **C4.5**, and on most problems the two algorithms have similar test error rates (within a few percent). It seems safe to say that **C4.5** is at least competitive with **Adaboost**. Certainly a gap in performance of the order suggested by the theory is not present.

What, then, is missing in the theoretical results, or more accurately, how have we been misinterpreting them? In our discussion so far, we have implicitly assumed that the advantage γ in the WHA remains constant (or is at least lower bounded by a constant) on *all* distributions. In fact, for Theorems 2.1 and 2.2 and Equation (4) to hold, it is only necessary that the WHA hold on the sequence of filtered distributions $P_1, P_2, \ldots, P_N$ *actually generated* by the algorithm under consideration. Thus, there might be some problems for which the WHA holds for the sequence of filtered distributions created by **TopDown**, but not on the sequence created by **Adaboost**. On

[1] Throughout this section, the splitting functions F used in the implementations of **Adaboost**$_F$ and **TopDown**$_F$ is the same as that used by **C4.5**, so we drop the subscript F.

such problems it would be perfectly consistent with the theoretical results for **TopDown** to outperform **Adaboost**. Furthermore, on real problems even this weakened version of the WHA is a rather unrealistic expectation: we should expect the best advantage γ_i that can be obtained within F on the filtered distribution P_i to be a *varying, algorithm-dependent and problem-dependent quantity*. We call the resulting sequence $\gamma_1, \gamma_2, \ldots$ the *advantage sequence* of the algorithm [2].

For example, in **Adaboost** we have already suggested that the successive filtered distributions are becoming "harder" as more weight is given to training examples on which previous hypotheses have erred. Thus, for the filtered distributions of **Adaboost**, we expect the advantage sequence to be decreasing. For **TopDown**, successive filtered distributions do not focus on hard examples, but form increasingly refined partitions of the input space. These partitions eventually become sufficiently fine that very simple functions can approximate the target function well within a cell. Thus, we expect the advantage sequence to increase, or at least remain roughly constant.

The key point is that even though for the same advantage sequence the bound for **Adaboost** is much better than that for **TopDown**, in practice the advantage sequences for the two algorithms are quite different. This could explain the discrepancy between the experimental parity of the algorithms and their theoretical disparity. Experimental results supporting this hypothesis are the focus of this section.

4.1 Methods

To measure the advantage sequence for **TopDown** experimentally, we need to compute the advantage of each split. Using the notation of Section 3, consider a candidate split of a decision tree node labeled by the function h, with a fraction q of positive examples at the parent and fractions p and r of positive examples at the two children. Let τ be the fraction of examples from the parent going to the r child. Kearns and Mansour [12] show that on the balanced distribution at the parent, h has the advantage[3]

$$\left| \frac{\tau}{2} \left(\frac{r}{q} - \frac{1-r}{1-q} \right) \right| \tag{7}$$

over random guessing (that is, the error of h on the balanced distribution is 1/2 minus the given expression). Thus, like the information gain, the advantage is entirely determined by the split parameters. We modified **C4.5** to compute the advantage at each split, and to record these advantages in the order that the splits would be generated by **TopDown**.

We measured the advantage sequence of algorithms **TopDown** and **Adaboost** on three problems: an artificial decision list problem, the Pima Indians diabetes database, and the Australian credit screening database. The advantage sequence for **TopDown** on any particular run is very noisy, because when a node containing few examples is split, the advantage tends to be high. **TopDown** tends to switch frequently between splitting such sparse nodes and splitting less sparse nodes, so the advantage sequence oscillates. To remove this noise, we averaged the results of 20 trials. To generate the trials in the decision list problem, we generated 20 separate training sets of 500 examples each. For the other two problems, we performed a 20-fold cross-validation.

To ensure that this advantage sequence comparison was fair, we first ran **TopDown** to determine how many rounds it required to drive the training error to zero. We then ran **Adaboost** for the same number of rounds and compared the training and test set errors for the two algorithms. The purpose of this was not to determine which algorithm was better, but to double-check that the algorithms were achieving similar performance for the same number of rounds of boosting. Table 2 shows the results of this check. **TopDown** is actually able to achieve zero training set error in fewer rounds than **Adaboost**, but the test set errors are roughly comparable. (The test error rates for the *pruned* trees of **C4.5** (not shown) are slightly below those shown for the unpruned trees.)

4.2 Results

Figures 8, 9, and 10 show the averaged advantage sequences for the three problems. In all cases, there is a dramatic difference between the advantage sequences of **TopDown** and **Adaboost**. As predicted, **Adaboost** rapidly drives the advantage to zero, falling below 0.05 within 40 rounds on all three problems. In contrast, the advantage sequence for **TopDown** is lowest at the beginning and tends to increase (or at least hold steady) as more splits are performed.

This demonstrates that the γ values in Theorems 2.1 and 2.2 cannot be assumed to be the same. If we think of the desired training error ϵ in Theorems 2.1 and 2.2 as being a small fixed constant, the differing behavior of $1/\gamma$ for the two algorithms goes a long way towards explaining their competitive performance in spite of the disparate dependence on $1/\gamma$ in the two theorems.

Figure 11 gives a scatter plot of the value of γ at a

[2] Freund and Schapire [10] give a generalized version of Theorem 2.1 which expresses the training error of **Adaboost** in terms of the advantage sequence, rather than a fixed advantage γ. A similar generalization of Theorem 2.2 is possible, but is beyond our scope. In any case, the resulting bounds again exhibit a disparity similar to that of Theorems 2.1 and 2.2.

[3] Technically, this advantage may actually be achieved by the complement of h, but since the class of splitting functions used by **C4.5** is closed under complementation, the effect is identical.

Table 2: Comparison of **TopDown** and **Adaboost** for equal numbers of boosting rounds (means of 20 trials).

Problem	**TopDown** # rounds to 0 train error	**Adaboost** train error	**TopDown** test error	**Adaboost** test error
Decision List	218.0	9.4	9.14	13.8
Pima Diabetes	143.6	18.0	29.60	23.7
Australian Credit	74.3	9.9	19.50	15.0

node and the fraction w of the training set that reaches that node for the decision list experiment. Not surprisingly, when w is large, the advantages are smaller: while the partition of the input space induced by the tree is still coarse, it may be hard to find good splits. On the other hand, if we confine our attention to nodes having at least 2% of the training data (which means at least 100 examples for this problem), we still see many large values for the advantage. This rules out the possibility that **C4.5** is only enjoying large advantages when there are a handful of examples at a node.

Finally, Figure 12 gives a scatter plot of the advantage versus the information gain to show that the advantage is measuring something quite different from the information gain. The information gain places bounds on the possible values for γ, but for any fixed value of the information gain, γ takes on a wide range of values. The mere fact that this scatter plot does not lie on a one-dimensional curve demonstrates that the advantage sequence measures something fundamentally different than the information gain.

4.3 Discussion

The Weak Hypothesis Assumption is a different way of defining inductive biases than previous theoretical frameworks. In the PAC model (and in the Bayesian and MDL frameworks), the bias is expressed in terms of assumptions about the syntactic representation, complexity, or prior probability of the target function. The WHA makes no explicit assumptions of this kind. Instead, it makes the incremental, procedural assumption that at each round of the boosting process, some $h \in F$ will have an advantage over random guessing.

Can our experimental results give us any intuition about what the WHA means for **TopDown** and **Adaboost**? Figures 8, 9, and 10 show that the advantage for **TopDown** is smallest near the root of the tree. Hence, for **C4.5**, the WHA is most likely to be violated early in the tree-growing process. In other words, for **C4.5**, the WHA asserts that even at the root, a split will have an advantage over random guessing. If we think about cases where this is violated (e.g., parity), these are precisely cases where **C4.5** fails.

For **Adaboost**, the advantages are small later in the boosting process, so the WHA is most dubious after several rounds of boosting have been performed. In other words, for **Adaboost**, after the "main effects" have been identified, the WHA asserts that **Adaboost** will still be able to make progress.

We can conclude that although making the WHA provide insights into the behavior and performance of both **Adaboost** and **C4.5**, the WHA is asserting fundamentally different things for the two algorithms.

References

[1] Yvonne M. M. Bishop, Stephen E. Fienberg, and Paul W. Holland. *Discrete Multivariate Analysis: Theory and practice*. MIT Press, Cambridge, MA, 1977.

[2] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*. ACM Press, New York, NY, 1994.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[4] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[6] N. H. Bshouty. Exact learning via the monotone theory. In *Proceedings of the 34th IEEE Symposium on the Foundations of Computer Science*, pages 302–311. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[7] T. G. Dietterich. Proper statistical tests for comparing supervised classification learning algorithms. Technical report, Department of Computer Science, Oregon State University, Corvallis, OR, 1996.

[8] Y. Freund and R. Schapire. Some experiments with a new boosting algorithm. These proceedings.

[9] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.

[10] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Second European Conference on Computational Learning Theory*, pages 23–37. Springer-Verlag, 1995.

[11] G. E. Hinton, R. M. Neal, R. Tibshirani, M. Revow, C. E. Rasmussen, D. van Camp, R. Kustra, and Z. Ghahramani. Assessing learning procedures using DELVE. Technical report, University of Toronto, 1995.

[12] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*. ACM Press, New York, NY, 1996. To appear.

[13] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. In *Proc. of the 23rd Symposium on Theory of Computing*, pages 455–464. ACM Press, New York, NY, 1991.

[14] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989.

[15] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases [machine-readable data repository]. Technical report, University of California, Irvine, 1994.

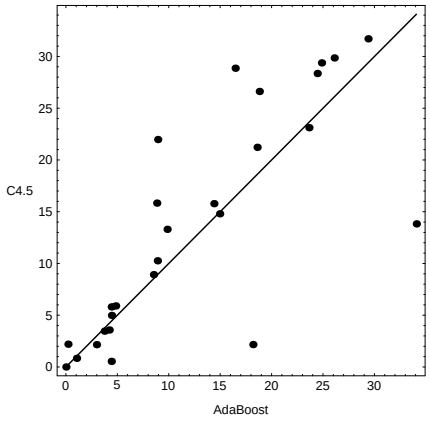

Figure 7: Scatter plot of test errors for **Adaboost** and **C4.5** on 27 problems from the UCI data repository [8]. Points below the diagonal indicate superior performance for **C4.5**, while points above the diagonal indicate superior performance for **Adaboost**.

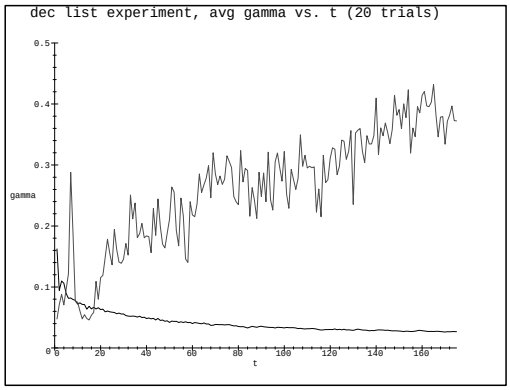

Figure 8: Plot of advantage γ_t on filtered distribution P_t vs. t for the unpruned tree of **C4.5** (top plot) and for **Adaboost** (bottom plot) on the artificial decision list learning problem.

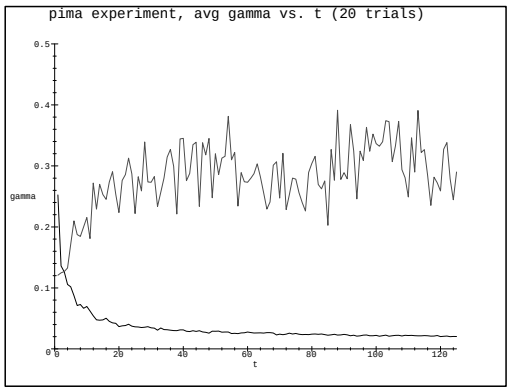

Figure 9: Plot of advantage γ_t on filtered distribution P_t vs. t for the unpruned tree of **C4.5** (top plot) and for **Adaboost** (bottom plot) on the Pima Indians diabetes problem.

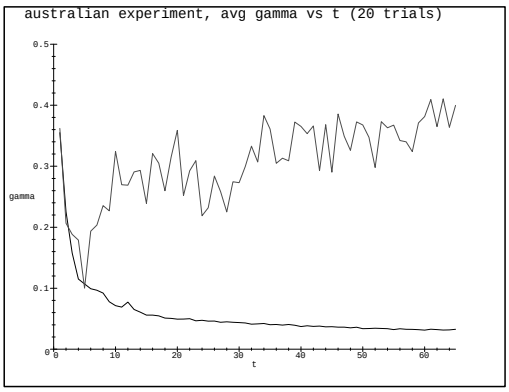

Figure 10: Plot of advantage γ_t on filtered distribution P_t vs. t for the unpruned tree of **C4.5** (top plot) and for **Adaboost** (bottom plot) on the Australian credit screening problem.

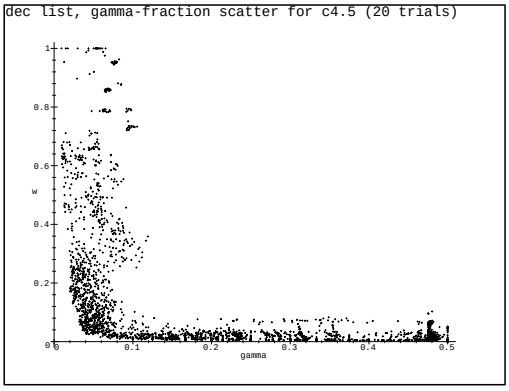

Figure 11: Scatter plot of advantage γ vs. weight w for **C4.5** on the decision list experiment, accumulated over all 20 trials. For each split made at node ℓ and labeled by the function h, we plot a point whose x coordinate is the advantage over random guessing that h has on the balanced distribution P'_ℓ, and whose y coordinate is $w(\ell)$, the fraction of the training sample reaching ℓ.

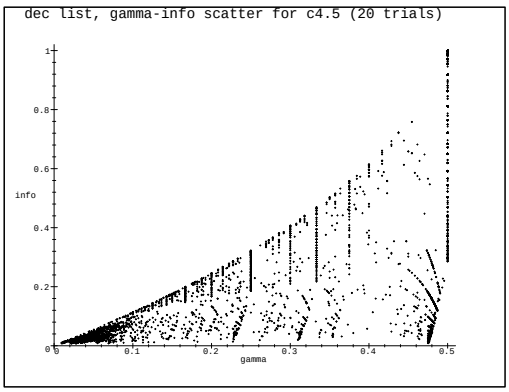

Figure 12: Scatter plot of advantage γ vs. information gain for **C4.5** on the decision list experiment, accumulated over all 20 trials. For each split made at node ℓ and labeled by the function h, we plot a point whose x coordinate is the advantage over random guessing that h has on the balanced distribution P'_ℓ, and whose y coordinate is the information gained by the split.