A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization

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

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

February 27, 1998

Abstract

In this work, we present a new bottom-up algorithm for decision tree pruning that is very efficient (requiring only a single pass through the given tree), and prove a strong performance guarantee for the generalization error of the resulting pruned tree. We work in the typical setting in which the given tree T may have been derived from the given training sample S, and thus may badly overfit S. In this setting, we give bounds on the amount of additional generalization error that our pruning suffers compared to the *optimal* pruning of T. More generally, our results show that if there is a pruning of T with small error, and whose size is small compared to $|S|$, then our algorithm will find a pruning whose error is not much larger. This style of result has been called an *index of resolvability* result by Barron and Cover in the context of density estimation.

A novel feature of our algorithm is its *locality* — the decision to prune a subtree is based entirely on properties of that subtree and the sample reaching it. To analyze our algorithm, we develop tools of *local uniform convergence*, a generalization of the standard notion that may prove useful in other settings.

Keywords: Decision Trees, Pruning, Theoretical Analysis, Model Selection, Uniform Convergence

1 Introduction

We consider the common problem of finding a good pruning of a given decision tree T on the basis of sample data S. We work in a setting in which we do *not* assume the independence of T and S. In particular, we allow for the possibility that T was in fact constructed from S, perhaps by a standard greedy, top-down process as employed in the growth phases of the C4.5 and CART algorithms [8, 3]. Our interest here is in how one should best use the data S a *second* time to find a good *subtree* of T. Note that in the setting we imagine, T itself may badly overfit the data.

Our main result is a new and rather efficient pruning algorithm, and the proof of a strong performance guarantee for this algorithm (Theorems 5 and 6). Our algorithm uses the sample S to compute a subtree (pruning) of T whose *generalization* error can be related to that of the *best* pruning of T. More generally, the generalization error of our pruning is bounded by the *minimum* over all prunings T' of the generalization error $\epsilon(T')$ plus a "complexity penalty" that depends only on the size of T'. Thus, if there is a relatively small subtree of T with small error, our algorithm enjoys a strong performance guarantee. This type of guarantee is fairly common in the model selection literature, and is sometimes referred to as an *index of resolvability* guarantee [1]. (It is also similar to the types of results stated in the literature on combining "experts" [4], although the interest there is not in generalization error but on-line prediction.) Our algorithm is a simple, bottom-up algorithm that performs a single pass over the tree T; hence its running time is linear in $size(T)$. The only information our algorithm needs for this bottom-up pass is, for each node in T, the depth and the number of positive and negative examples reaching the node. This information is typically available from the construction of the tree, or can be computed in time $O(|S| depth(T))$.

An important aspect of our algorithm is its *locality*. Roughly speaking, this means that the decision to prune or not prune a particular subtree during the execution is based entirely on properties of that subtree and the sample that reaches it. We argue that locality is an intuitively sensible property for a pruning algorithm to have, and the analysis of our algorithm requires us to develop the notion of *local uniform convergence*, a generalization of the standard notion of uniform convergence, and a tool that we believe may prove useful in other settings.

There are a number of previous efforts related to our results, which we only have space to discuss briefly here; more detailed comparisons will be given in the full paper. First of all, our pruning algorithm is closely related to one proposed by Mansour [7], who emphasized the locality property and gave primarily experimental results, but was not able to bound the generalization error of the resulting pruned tree.

Helmbold and Schapire [4] gave an efficient algorithm for *predicting* nearly as well as the best pruning of a given tree. However, this algorithm differs from ours in a number of important ways. First of all, it *cannot* be directly applied to the same data set that was used to derive the given tree in order to obtain a good pruning — the predictive power is only on a "fresh" or held-out data set. (A standard transformation of their algorithm can be used on the original data set, but results in a considerably less efficient algorithm, as it requires many executions of the algorithm.) Second, it does not actually find a good pruning of the given tree, but a weighted combination of prunings. However, in the on-line prediction model of learning, their result is quite strong. Here we study the typical batch model in which we may not assume independence of our tree and data set.

The use of dynamic programming for pruning was already suggested in the original book on CART [3] in order to minimize a weighted sum of the observed error and the size of the pruning. Bohanec and Bratko [2] showed that it is possible to compute in quadratic time the subtree of a given tree that minimizes the training error while obeying a specified size bound. By combining this observation with the ideas of structural risk minimization [10], it is possible to derive a polynomial-time algorithm for our setting with error guarantees quite similar to those we will give for our algorithm. However, this algorithm would be considerably less efficient than the one we shall present.

Finally, our ideas are certainly influenced by the many single-pass, bottom-up pruning heuristics in wide use in experimental machine learning, including that used by C4.5 [8]. While we do not know how to prove strong error guarantees for these heuristics, our current results provide some justification for them, and suggest specific modifications that yield fast, practical and principled methods for pruning with proven error guarantees. Combined with earlier results proving non-trivial performance guarantees for the common greedy, top-down growth heuristics in the model of boosting [5], it is fair to say that there is now a solid theoretical basis for both the top-down and bottom-up passes of many standard decision tree learning algorithms.

2 Framework and Preliminaries

We consider decision trees over an input domain X. Each such tree has binary tests at each internal node, where each test is chosen from a class $\mathcal{T}$ of predicates over X. We use $\textsc{trees}(\mathcal{T}, d)$ to denote the class of all binary trees with tests from $\mathcal{T}$ and at most d internal nodes, and leaves labeled with 0 or 1.

We will also need notation to identify paths in a decision tree. Thus, we use $\textsc{paths}(\mathcal{T}, \ell)$ to denote the class of all conjunctions of at most ℓ predicates from $\mathcal{T}$. Clearly, if v is a node in a decision tree $T \in \textsc{trees}(\mathcal{T}, d)$, then we may associate with v a predicate $reach_v \in \textsc{paths}(\mathcal{T}, d)$, which is simply the conjunction of the predicates along the path from the root to v in T. Thus, for any input $x \in X$, $reach_v(x) = 1$ if and only if the path defined by x in T passes through v.

Given a node v in T, we let T_v denote the subtree of T that is rooted at v, and for any probability distribution P over X, we let P_v denote the distribution induced by P on just those x satisfying $reach_v(x) = 1$.

In our framework, there is an unknown distribution P over X and an unknown *target function* f over X. We are given a sample S of m pairs $\langle x_i, f(x_i)\rangle$, where each x_i is drawn independently according to P. We are also given a tree $T = T(S)$ that may have been built from the sample S. Now for f and T fixed, for any distribution P, we define the *generalization error* $\epsilon(T) = \epsilon_P(T) = \mathbf{Pr}_P[T(x) \neq f(x)]$, and also the *training error* $\hat{\epsilon}(T) = \hat{\epsilon}_S(T) = (1/m)\sum_{i=1}^{m} I[T(x) \neq f(x)]$, where I is the indicator function. In this notation, for any node v in T, we can define the *local* generalization error $\epsilon_v = \epsilon_{P_v}(T_v)$ and the local training error $\hat{\epsilon}_v = (1/|S_v|)\sum_{x\in S_v} I[T(x) \neq f(x)]$, where S_v is the set of all $x \in S$ satisfying $reach_v(x) = 1$. We will also need to refer to the local errors incurred by deleting the subtree T_v and replacing it by a leaf with the majority label of the examples reaching v. Thus, we use $\epsilon_v(\emptyset)$ to denote $\min\{\mathbf{Pr}_{P_v}[f(x) = 0], \mathbf{Pr}_{P_v}[f(x) = 1]\}$; this is exactly the error, with respect to P_v, of the optimal constant function (leaf) 0 or 1. Similarly, we will use $\hat{\epsilon}_v(\emptyset)$ to denote $(1/|S_v|)\min\{|\{x \in S_v : f(x) = 0\}|, |\{x \in S_v : f(x) = 1\}|\}$, which is the observed local error incurred by replacing T_v by the best leaf.

As we have mentioned in the introduction, we make no assumptions on f, and our goal is not to "learn" f in the standard sense of, say, fitting a decision tree to the data and hoping that it generalizes well. Here we limit our attention to the problem of *pruning* a *given* decision tree. Thus, we assume that we are given as input the sample S and a particular, fixed tree T, with the goal of finding a *pruning* of T with near-optimal generalization.

It is important to specify what we mean by a pruning of T, since allowing different pruning operations clearly can result in different classes of trees that can be obtained from T. We let PRUNINGS(T) denote the class of all subtrees of T, that is, the class of all trees that can be obtained from T by specifying nodes $v_1, \ldots, v_k$ in T and then deleting from T the subtrees $T_{v_1}, \ldots, T_{v_k}$ rooted at those nodes. The allowed operation is that of deleting any subtree from the current tree; PRUNINGS(T) is exactly the class of trees that can be obtained from T by any sequence of such operations. Thus, any non-empty tree in PRUNINGS(T) shares the same root as T, and can be "superimposed" on T. In particular, we are not allowing "surgical" operations such as the replacement of an internal node by its left or right subtree [8]. Nevertheless, the class PRUNINGS(T) contains an exponential number of subtrees of T, and our goal will be find a tree in PRUNINGS(T) with close to the smallest generalization error.

Let us again emphasize that we do *not* assume any "independence" between the given tree T and the sample S — indeed, the likely scenario is that T was built *using* S. Formally, we are given a pair (S, T) in which we allow $T = T(S)$. We are imagining the common scenario in which the sample S is to be used *twice* — once for top-down growth of T using a heuristic such as those used by C4.5 or CART, and now again to find a good subtree of T. If one assumes that S is a "fresh" or held-out sample (that is, drawn *separately* from the sample used to construct T), the problem becomes easier in some ways, since one can then use the observed error on S as an approximate proxy for the generalization error of any tree in PRUNINGS(T). There is a trade-off that renders the two scenarios incomparable in general [6]: by using a hold-out set for the pruning phase, we gain the independence of the sample from the given tree T, but at the price of having "wasted" some potentially valuable data for the training (construction) of T; whereas in our setting, we waste no data, but cannot exploit independence of T and S.

In the hold-out setting, a good algorithm is one that chooses the tree in PRUNINGS(T) that minimizes the error on S (which can be computed in polynomial time via a dynamic programming approach [2]), and fairly general performance guarantees can be shown [6] that necessarily weaken as the hold-out set becomes a smaller fraction of the original data sample.

3 Description of the Pruning Algorithm

We begin with a detailed description of the pruning algorithm, which is given the random sample S and a tree $T = T(S)$ as input. The high-level structure of the algorithm is quite straightforward: the algorithm makes a single "bottom-up" pass through T, and decides for every node v whether to leave the subtree currently rooted at v in place (for the moment), or whether to delete this subtree. More precisely, imagine that we place a marker at each leaf of T, and for any node v in T, let MARKERS(v) denote the set of markers in the subtree T_v rooted at v. When *all* of the markers in MARKERS(v) have arrived at v, our algorithm will then (and only then) consider whether or not to delete the subtree then rooted at v; the algorithm then passes all of these markers to its parent. Thus, the algorithm only considers pruning at a node v once it has first considered pruning at all nodes below v; this simply formalizes the standard notion of "bottom-up" processing.

Two observations are in order here. First, the algorithm considers a pruning operation only once at each node v of T, at the moment when all of MARKERS(v) resides at v. Second, the subtree rooted at v when all of MARKERS(v) reside at v may be *different* than T_v, because parts of T_v may have been deleted as markers were being passed up towards v. We thus introduce the notation T_v^* to denote the subtree that is rooted at v when all of MARKERS(v) resides at v. It is T_v^* that our algorithm must decide whether to prune, and T_v^* is defined by the operation of the algorithm itself. We will use T^* to denote the final pruning of T output by our algorithm.

It remains only to describe how our algorithm decides whether or not to prune T_v^*. For this we need some additional notation. We define $m_v = |S_v|$, and we let s_v denote the number of nodes in T_v^*, and ℓ_v be the depth of the node v in T. Recall that $\hat{\epsilon}_v(T_v^*)$ is the fraction of errors T_v^* makes on the local sample S_v, and $\hat{\epsilon}_v(\emptyset)$ is the fraction of errors the best leaf makes on S_v. Then our algorithm will replace T_v^* by this best leaf if and only if

$$\hat{\epsilon}_v(T_v^*) + \alpha(m_v, s_v, \ell_v, \delta) \geq \hat{\epsilon}_v(\emptyset) \tag{1}$$

where $\delta \in [0,1]$ is a confidence parameter. The exact choice of $\alpha(m_v, s_v, \ell_v, \delta)$ will depend on the setting, but in all cases can be thought of as a penalty for the complexity of the subtree T_v^*. Let us first consider the case in which the class $\mathcal{T}$ of testing functions is finite, in which case the class of possible path predicates $\text{PATHS}(\mathcal{T}, \ell_v)$ leading to v and the class of possible subtrees $\text{TREES}(\mathcal{T}, s_v)$ rooted at v are also finite. In this case, we would choose

$$\alpha(m_v, s_v, \ell_v, \delta) = c\sqrt{\frac{\log(|\text{PATHS}(\mathcal{T}, \ell_v)|) + \log(|\text{TREES}(\mathcal{T}, s_v)| + \log(m/\delta)}{m_v}} \tag{2}$$

for some constant $c > 1$. Perhaps the most natural and common special case of this finite-cardinality setting is that in which the input space X is the boolean hypercube $\{0,1\}^n$, and the test class $\mathcal{T}$ contains just the n single-variable tests x_i. These are the kinds of tests allowed in the vanilla C4.5 and CART packages, and since $|\text{PATHS}(\mathcal{T}, \ell)| \leq n^\ell$ and $|\text{TREES}(\mathcal{T}, s)| \leq (an)^s$ for some constant a, Equation (2) specializes to

$$\alpha(m_v, s_v, \ell_v, \delta) = c'\sqrt{\frac{(\ell_v + s_v)\log(n) + \log(m/\delta)}{m_v}} \tag{3}$$

for some constant $c' > 1$. To simplify the exposition and to make it more concrete, we will work this particular choice of $\mathcal{T}$ in most of our proofs, but specifically point out how the analysis changes for the case of infinite $\mathcal{T}$, where the pruning rule is given by choosing

$$\alpha(m_v, s_v, \ell_v, \delta) = c\sqrt{\frac{(d_{\ell_v} + d_{s_v})\log(2m) + \log(m/\delta)}{m_v}} \tag{4}$$

for some constant $c > 1$, where d_{ℓ_v} and d_{s_v} are the VC dimensions of the classes $\text{PATHS}(\mathcal{T}, \ell_v)$ and $\text{TREES}(\mathcal{T}, s_v)$, respectively.

Let us first provide some brief intuition behind our algorithm, which will serve as motivation for the ensuing analysis as well. At each node v, our algorithm considers whether to leave the current subtree T_v^* or to delete it. The basis for this comparison must clearly make use of the sample S provided.

Beyond this observation, a number of ways of comparing T_v^* to the best leaf are possible. For instance, we could simply prefer whichever of T_v^* and the best leaf makes the smaller number of mistakes on S_v. This is clearly a poor idea, since T_v^* cannot do worse than the best leaf (assuming majority labels on the leaves of T_v^*), and may do considerably better — but generalize poorly compared to the best leaf due to overfitting. Thus, it seems we should penalize T_v^* for its complexity, which is exactly the role of the additive term $\alpha(m_v, s_v, \ell_v, \delta)$ above.

One important subtlety of our algorithm is the fact that the comparison between T_v^* and the best leaf is being made entirely on the basis of the *local* reduction to the observed error. That is, the comparison depends on S_v and T_v^* only, and not on all of S and T. A reasonable alternative "global" comparison might compare the observed error of the entire tree, $\hat{\epsilon}(T^*)$, plus a penalty term that depends on $size(T^*)$, with the observed error of the entire tree with T_v^* pruned, $\hat{\epsilon}(T^* - T_v^*)$ (where $T^* - T_v^*$ is the tree after we prune at v), plus a penalty term that depends on $size(T^* - T_v^*)$. The important difference between this global algorithm and ours is that in the global algorithm, even when there is a large *absolute* difference in complexity between T_v^* and a leaf, this difference may be swamped by the fact that both are embedded in the much larger supertree T^* — that is, the difference is small *relative* to the complexity of T^*. This may cause a suboptimal insensitivity, leading to a propensity to leave large subtrees unpruned. Indeed, it is possible to construct examples in which the global approach leads to prunings strictly worse than those produced by our algorithm, and demonstrating that results as strong as we will give are not possible for the global method.

Our analysis proceeds as follows. We first need to argue that any time our algorithm chooses *not* to prune T_v^*, then (with high probability) this was in fact the "right" decision, in the sense that the current tree T^* would be degraded by deleting T_v^*. This allows us to establish that our final pruning will be a subtree of the optimal pruning, so our only source of additional error results from those subtrees of this optimal pruning that we deleted. A careful amortized analysis allows us to bound this additional error by a quantity related to the size of the optimal pruning. This line of argument establishes a relationship between the error of our pruning and that of the optimal pruning; a slight modification of the algorithm and a more involved analysis let us make a similar comparison to *any* pruning. This extension is important for cases in which there may be a pruning whose error is only slightly worse than that of the optimal pruning, but whose size is much smaller. In such a case our bounds are much better.

4 Local Uniform Convergence

In a standard uniform convergence results, we have a class of events (predicates), and we prove that the observed frequency of any event in the class does not differ much from its true probability. We would like to apply such results to events of the form "subtree T_v^* makes an error on x", but do not wish to take what is perhaps most obvious approach towards doing so. The reason is that we want to examine this event *conditioned* on the event that x reaches v, and obviously this conditioning event differs for every v. One approach would be to redefine the class of events of interest to include the conditioning events, that is, to look at events of the form "x satisfies $reach_v(x)$ and T_v^* makes an error on x" for all possible $reach_v(x)$ and T_v^*. It turns out that this approach would result in final bounds on our performance that are significantly worse than what we will obtain. What we really want is the rather natural notion of *local uniform convergence*: for any conditioning event c in a class C, and any event e in a class E, we would like to relate the observed frequency of e *restricted to the subsample satisfying* c to the true probability of e on the *distribution conditioned on* c; and clearly the accuracy of this observed frequency will depend not on the overall sample size, but on the number of examples satisfying the conditioning event c. Such a relationship is given by the next two theorems, which treat the cases of finite classes and infinite classes separately.

Lemma 1 *Let C and H be finite classes of boolean functions over X, let f be a target boolean function over x, and let P be a probability distribution over X. For any $c \in C$ and $h \in H$, let $\epsilon_c(h) = \mathbf{Pr}_P[h(x) \neq f(x)|c(x) = 1]$, and for any labeled sample S of $f(x)$, let $\hat{\epsilon}_c(h)$ denote the fraction of points in S_c on which h errs, where $S_c = \{x \in S : c(x) = 1\}$. Then the probability that there exists a $c \in C$ and an $h \in H$ such that*

$$|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \sqrt{\frac{\log(|C|) + \log(|H|) + \log(1/\delta)}{m_c}} \tag{5}$$

is at most δ, where $m_c = |S_c|$.

Proof: Let us fix $c \in C$ and $h \in H$. For these fixed choices, we have for any value λ

$$\mathbf{Pr}_P[|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \lambda] = \mathbf{E}_{m_c}[\mathbf{Pr}_{S_c}[|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \lambda]]. \tag{6}$$

Here the expectation is over the distribution on values of m_c induced by P, and the distribution on S_c is over samples of size m_c (which is fixed inside the expectation) drawn according to P_c (the distribution P conditioned on c being 1). Since m_c is fixed, by standard Chernoff bounds we have

$$\mathbf{Pr}_{S_c}[|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \lambda] \leq e^{-\lambda^2 m_c} \tag{7}$$

giving the bound

$$\mathbf{Pr}_P[|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \lambda] \leq \mathbf{E}_{m_c}[e^{-\lambda^2 m_c}]. \tag{8}$$

If we choose

$$\lambda = \sqrt{\frac{\log(|C|) + \log(|H|) + \log(1/\delta)}{m_c}} \tag{9}$$

then $e^{-\lambda^2 m_c} = \delta/(|C||H|)$, which is a constant independent of m_c and thus can be moved outside the expectation. By appealing to the union bound ($\mathbf{Pr}[A \vee B] \leq \mathbf{Pr}[A] + \mathbf{Pr}[B]$), the probability that there is *some* c and h such that $|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \lambda$ is at most $|C||H|(\delta/(|C||H|)) = \delta$, as desired. $\Box$

Our use of Lemma 1 will be straightforward. Suppose we are considering some node v in a decision tree T at depth ℓ_v, and with a subtree T_v^* of size s_v rooted at v. Then we will appeal to the lemma choosing the conditioning class C to be the class $\mathrm{PATHS}(\mathcal{T}, \ell_v)$, choosing H to be $\mathrm{TREES}(\mathcal{T}, s_v)$, and choosing δ to be δ'/m^2. In this case, the local complexity penalty $\alpha(\ell_v, s_v, m_v, \delta)$ in Equation (2) and the deviation λ in Equation (9) coincide, and thus we can assert that with probability $1 - \delta'/m^2$ there is no leaf of depth ℓ_v and subtree of size s_v such that the local observed error of the subtree deviates by more than $\alpha(m_v, s_v, \ell_v, \delta)$ from the local true error. By summing over all m^2 choices for ℓ_v and s_v, we obtain an overall bound of δ on the failure probability.

In other words, if we limit our attention to the *local* errors ϵ_v (generalization) and $\hat{\epsilon}_v$ (observed), then with high probability we can assert that they will be within an amount (namely, $\alpha(m_v, s_v, \ell_v, \delta)$) that depends only on *local* quantities: the local sample size m_v, the length ℓ_v of the path leading to v, and the size of the subtree rooted at v.

A more complicated argument is needed to prove local uniform convergence for the case of infinite classes.

Lemma 2 *Let C and H be classes of boolean functions over X, let f be a target boolean function over x, and let P be a probability distribution over X. For any $c \in C$ and $h \in H$, let $\epsilon_c(h) = \mathbf{Pr}_P[h(x) \neq f(x)|c(x) = 1]$, and for any labeled sample S of $f(x)$, let $\hat{\epsilon}_c(h)$ denote the fraction of points in S_c on which h errs, where $S_c = \{x \in S : c(x) = 1\}$. Then the probability that there exists a $c \in C$ and an $h \in H$ such that*

$$|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \sqrt{\frac{(d_C + d_H)\log(2m) + \log(1/\delta)}{m_c}} \tag{10}$$

is at most δ, where $m_c = |S_c|$, and d_C and d_H are the VC dimensions of C and H, respectively.

Proof: (Sketch) The proof closely follows the "two-sample trick" proof for the classical VC theorem [9], with an important variation. Intuitively, we introduce a "nested two-sample trick", since we need to apply the idea twice — once for C, and again for H.

As in the classical proof, we define two events, but now they are "local" events. Event $A(S)$ is that in a random sample S of m examples, there exists a $c \in C$ and an $h \in H$ such that $|\epsilon_c(h) - \hat{\epsilon}_c(h)| \geq \lambda$. Event $B(S, S')$ is that in a random sample $S \cup S'$ of $2m$ examples, there exists a $c \in C$ and an $h \in H$ such that $|\hat{\epsilon}_c(h) - \hat{\epsilon}'_c(h)| \geq \lambda/2$, where $\hat{\epsilon}_c(h)$ and $\hat{\epsilon}'_c(h)$ denote the observed local error of h on S and S', respectively.

We use the fact that

$$\begin{aligned}\mathbf{Pr}_S[A(S)] &= \mathbf{Pr}_{S,S'}[A(S)] &(11)\\ &= \mathbf{Pr}_{S,S'}[A(S)\wedge B(S,S')]/\mathbf{Pr}_{S,S'}[B(S,S')|A(S)] &(12)\end{aligned}$$

Clearly, $\mathbf{Pr}_{S,S'}[A(S)\wedge B(S,S')] \leq \mathbf{Pr}_{S,S'}[B(S,S')]$. We also have the inequality $\mathbf{Pr}_{S,S'}[B(S,S')|A(S)] \geq 1/2$. Therefore, $\mathbf{Pr}_{S,S'}[A(S)] \leq 2\mathbf{Pr}_{S,S'}[B(S,S')]$, and we can concentrate on bounding the probability of event B.

Let us first consider a *fixed* set of $2m$ inputs $x_1, \ldots x_{2m}$. The number of possible subsets of this set induced by taking intersections with sets in C is at most $\Phi_C(2m)$, where Φ_C is the dichotomy counting functions of classical VC analysis. Let us fix a $c \in C$, and consider the subset S_c of $x_1, \ldots, x_{2m}$ that fall in c; let $m_c = |S_c|$. Now consider all possible labelings of S_c by the concept class H; there are at most $\Phi_H(m_c) \leq \Phi_H(2m)$ such labelings. Let us now also fix one of these labelings, by fixing some $h \in H$.

Now both $c \in C$ and $h \in H$ are fixed. Consider splitting S_c randomly into two subsets of size $m_c/2$ each, S_c^1 and S_c^2. For event B to hold, we need the difference between the observed errors of h on S_c^1 and S_c^2 to be at least $\lambda/2$. It can be shown that this will occur with probability at most $e^{-\lambda^2 m_c/12}$, where the probability is taken only over the random partitioning of S_c. Now if we choose

$$\lambda = \sqrt{\frac{(d_C + d_H)\log(2m) + \log(1/\delta)}{m_c}} \tag{13}$$

then $e^{-\lambda^2 m_c/12} = (1/(2m)^{d_C})(1/(2m)^{d_H})\delta$, which is independent of m_c. We can then bound the probability that this event occurs for *some* c and h by summing this bound over all possible subsets S_c, and all possible labelings of S_c by functions in H, giving a bound of $\Phi_C(2m)\Phi_H(2m)(1/m^{d_C})(1/m^{d_H})\delta$. Using the fact that $\Phi_C(m) \leq m^{d_C}$ and $\Phi_H(m) \leq m^{d_H}$ yields an overall bound of δ, as desired. □

5 Analysis of the Pruning Algorithm

In this section, we apply the tools of local uniform convergence to analyze the pruning algorithm given in Section 3. As mentioned earlier, for simplicity in exposition, we will limit our attention to the common case in which X is the boolean hypercube $\{0,1\}^n$ and the class $\mathcal{T}$ of allowed node tests is just the input variables x_i, in which case the pruning rule used by our bottom-up algorithm is that given by Equation (3). However, it should be clear how the analysis easily generalizes to the more general algorithms given by Equations (2) and (4).

We shall first give an analysis that compares the generalization error of the pruning T^* produced by our algorithm from S and T to the generalization error of T_{opt}, the pruning of T that minimizes the generalization error. Recall that we use T_v^* to denote the subtree that is rooted at node v of T *at the time our algorithm decides whether or not to prune at* v, which may be a subtree of T_v due to prunings that have already taken place below v.

We will show that $\epsilon(T^*)$ is larger than $\epsilon_{opt} = \epsilon(T_{opt})$ by an amount that can be bounded by the size s_{opt} and depth ℓ_{opt} of T_{opt}. Thus, if there is a reasonably small subtree of T with small generalization error, our algorithm will produce a pruning with small generalization error as well. In Section 6, we will improve our analysis to compare the error of T^* to that of *any* pruning, and provide a discussion of situations in which this result may be considerably more powerful than our initial comparison to T_{opt} alone.

For the analysis, it will be convenient to introduce the notation

$$r_v = (\ell_v + s_v)\log(n) + \log(m/\delta) \tag{14}$$

for any node v, where ℓ_v is the depth of v in T, and s_v is the size of T_v^*. In this notation, the penalty $\alpha(m_v, s_v, \ell_v, \delta)$ given by Equation (3) is simply a constant (that we ignore in the analysis for ease of exposition) times $\sqrt{r_v/m_v}$. (We assume that $\sqrt{r_v/m_v} \leq 1$, since a penalty which is larger than 1 can be modified to a penalty of 1 without changing the results.)

Lemma 3 *With probability at least* $1-\delta$ *over the draw of the input sample* S, T^* *is a subtree of* T_{opt}.

Proof: Consider any node v that is a leaf in T_{opt}. It suffices to argue that our algorithm would choose to prune T_v^*, the subtree that remains at v when our algorithm reaches v. By Equation (3), our algorithm would *fail* to prune T_v^* only if $\hat{\epsilon}_v(\emptyset)$ exceeded $\hat{\epsilon}_v(T_v^*)$ by at least the amount $\alpha(m_v, s_v, \ell_v, \delta)$, in which case Lemma 1 ensures that $\epsilon_v(T_v^*) < \epsilon_v(\emptyset)$ with high probability. In other words, if our algorithm fails to prune T_v^*, then T_{opt} would have smaller generalization error by including T_v^* rather than making v a leaf. This contradicts the optimality of T_{opt}. □

Lemma 3 means that the only source of additional error of T^* compared to T_{opt} is through overpruning, not underpruning. Thus, for the purposes of our analysis, we can imagine that our algorithm is actually run on T_{opt} rather than the original input tree T (that is, the algorithm is initialized starting at the leaves of T_{opt}, since we know that the algorithm will prune everything below this frontier).

Let $V = \{v_1, \ldots, v_t\}$ be the sequence of nodes in T_{opt} at which the algorithm chooses to prune the subtree $T_{v_i}^*$ rather than to leave it in place; note that $t \leq s_{opt}$. Then we may express the additional generalization error $\epsilon(T^*) - \epsilon_{opt}$ as

$$\epsilon(T^*) - \epsilon_{opt} = \sum_{i=1}^{t} (\epsilon_{v_i}(\emptyset) - \epsilon_{v_i}(T_{v_i}^*)) p_{v_i} \tag{15}$$

where p_{v_i} is the probability under the input distribution P of reaching node v_i, that is, the probability of satisfying the path predicate $reach_{v_i}$. Each term in the summation of Equation (15) simply gives the change to the *global* error incurred by pruning $T^*_{v_i}$, expressed in terms of the *local* errors. Clearly the additional error of T^* is the sum of all such changes.

Now we may write

$$\begin{aligned}
\epsilon(T^*) - \epsilon_{opt} &\leq \sum_{i=1}^{t} \Big(|\epsilon_{v_i}(\emptyset) - \hat{\epsilon}_{v_i}(\emptyset)| + |\hat{\epsilon}_{v_i}(\emptyset) - \hat{\epsilon}_{v_i}(T^*_{v_i})| \\
&\qquad + |\hat{\epsilon}_{v_i}(T^*_{v_i}) - \epsilon_{v_i}(T^*_{v_i}|\Big) p_{v_i} &(16)\\
&\leq \sum_{i=1}^{t} \left(\sqrt{\frac{(\ell_{v_i}+1)\log(n) + \log(m/\delta)}{m_{v_i}}} + \alpha(m_{v_i}, s_{v_i}, \ell_{v_i}, \delta) \right. \\
&\qquad \left. + \sqrt{\frac{(\ell_{v_i}+s_{v_i})\log(n) + \log(m/\delta)}{m_{v_i}}} \right) p_{v_i} &(17)\\
&\leq 4\sum_{i=1}^{t} \left(\sqrt{\frac{r_{v_i}}{m_{v_i}}} \right) p_{v_i}. &(18)
\end{aligned}$$

The first inequality comes from the triangle inequality. The second inequality uses two invocations of Lemma 1, and the fact that our algorithm directly compares $\hat{\epsilon}_{v_i}(\emptyset)$ and $\hat{\epsilon}_{v_i}(T^*_{v_i})$, and prunes only when they differ by less than $\alpha(m_{v_i}, s_{v_i}, \ell_{v_i}, \delta)$.

Thus, we would like to bound the sum $\Delta = \sum_{i=1}^{t}(\sqrt{r_{v_i}/m_{v_i}})p_{v_i}$. The leverage we will eventually use is the fact that $\sum_{i=1}^{t} r_{v_i}$ can be bounded by quantities involving only the tree T_{opt}, since all of the $T^*_{v_i}$ are disjoint subtrees of T_{opt}. First it will be convenient to break this sum into two sums — one involving just those terms for which p_{v_i} is "small", and the other involving just those terms for which p_{v_i} is "large". The advantage is that for the large p_{v_i}, we can relate p_{v_i} to its empirical estimate $\hat{p}_{v_i} = m_{v_i}/m$, as evidenced by the following lemma.

Lemma 4 *The probability, over the sample S, that there exists a node $v_i \in V$ such that $p_{v_i} > 12\log(t/\delta)/m$ but $p_{v_i} \geq 2\hat{p}_{v_i}$ is at most δ.*

Proof: We will use the relative Chernoff bound

$$\mathbf{Pr}[\hat{p}_{v_i} < (1-\gamma)p_{v_i}] \leq e^{mp\gamma^2/3} \tag{19}$$

which holds for any fixed v_i. By taking $\gamma = 1/2$ and applying the union bound, we obtain

$$\mathbf{Pr}[\exists v_i \in V : p_v \geq 2\hat{p}_v] \leq te^{-p_{v_i}m/12}. \tag{20}$$

Now we can use the assumed lower bound on p_{v_i} to bound the probability of the event by δ. $\square$

Let V' be the subset of V for which the lower bound $p_{v_i} > 12\log(t/\delta)/m$ holds. We divide the sum that describes Δ into two parts:

$$\Delta = \sum_{v_i \in V-V'} \left(\sqrt{r_{v_i}/m_{v_i}}\right) p_{v_i} + \sum_{v_i \in V'} \left(\sqrt{r_{v_i}/m_{v_i}}\right) p_{v_i} \tag{21}$$

The first summation is bounded by $12\log(s_{opt}/\delta)s_{opt}/m$, since $\sqrt{r_{v_i}/m_{v_i}}$ is at most 1, and $t \leq s_{opt}$.

For the second summation, we perform a maximization. By Lemma 4, with high probability we have that for every $v_i \in V'$, $p_{v_i} < 2\hat{p}_{v_i} = 2m_{v_i}/m$. Thus, with high probability we have

$$\begin{aligned}
\sum_{v_i \in V'} \sqrt{\frac{r_{v_i}}{m_{v_i}}} p_{v_i} &< \sum_{v_i \in V'} \sqrt{\frac{r_{v_i}}{m_{v_i}}} \left(2\frac{m_{v_i}}{m}\right) && (22)\\
&= \frac{2}{m} \sum_{v_i \in V'} \sqrt{r_{v_i}}\sqrt{m_{v_i}} && (23)\\
&\leq \frac{2}{m} \sqrt{\left(\sum_{v_i \in V'} r_{v_i}\right)\left(\sum_{v_i \in V'} m_{v_i}\right)}. && (24)
\end{aligned}$$

To bound this last expression, we first bound $\sum_{v_i \in V'} r_{v_i}$. Recall that

$$r_{v_i} = (\ell_{v_i} + s_{v_i})\log(n) + \log(m/\delta). \tag{25}$$

Since for any $v_i \in V'$, we have $\ell_{v_i} \leq \ell_{opt}$, we have that $\sum_{v_i \in V'} \ell_{v_i} \leq s_{opt}\ell_{opt}$, since $|V'| \leq t \leq s_{opt}$. Since the subtrees $T^*_{v_i}$ that we prune are disjoint and subsets of the optimal subtree T_{opt}, we have $\sum_{v_i \in V'} s_{v_i} \leq s_{opt}$. Thus

$$\sum_{v_i \in V'} r_i \leq s_{opt}((1+\ell_{opt})\log(n) + \log(m/\delta)). \tag{26}$$

To bound $\sum_{v_i \in V'} m_{v_i}$ in Equation (24), we observe that since the sets of examples that reach different nodes at the same level of the tree are disjoint, we have $\sum_{v_i \in V'} m_{v_i} \leq m\ell_{opt}$. Thus, with probability $1-\delta$, we obtain an overall bound

$$\begin{aligned}
\Delta &< 12\log(s_{opt}/\delta)\frac{s_{opt}}{m} \\
&\quad + \frac{2}{m}\sqrt{s_{opt}((1+\ell_{opt})\log(n) + \log(m/\delta))(m\ell_{opt})} && (27)\\
&= O\left(\left(\log(s_{opt}/\delta) + \ell_{opt}\sqrt{\log(n) + \log(m/\delta)}\right)\sqrt{\frac{s_{opt}}{m}}\right) && (28)
\end{aligned}$$

This gives the first of our main results.

Theorem 5 *Let S be a random sample of size m drawn according an unknown target function and input distribution. Let $T = T(S)$ be any decision tree, and let T^* denote the subtree of T output by our pruning algorithm on inputs S and T. Let ϵ_{opt} denote the smallest generalization error among all subtrees of T, and let s_{opt} and ℓ_{opt} denote the size and depth of the subtree achieving ϵ_{opt}. Then with probability $1-\delta$ over S,*

$$\epsilon(T^*) - \epsilon_{opt} = O\left(\left(\log(s_{opt}/\delta) + \ell_{opt}\sqrt{\log(n) + \log(m/\delta)}\right)\sqrt{\frac{s_{opt}}{m}}\right) \tag{29}$$

6 An Index of Resolvability Result

Roughly speaking, Theorem 5 ensures that the true error of the pruning found by our algorithm will be larger than that of the best possible pruning by an amount that is not much worse than $\sqrt{s_{opt}/m}$ (ignoring logarithmic and depth factors for simplicity). How good is this? Since we assume that T itself (and therefore, all subtrees of T) may have been constructed from the sample S, standard model selection analyses [10] indicate that ϵ_{opt} may be larger than the error of the best decision tree approximation to the target function by an amount growing like $\sqrt{s_{opt}/m}$. (Recall that ϵ_{opt} is only the error of the optimal *subtree of T* — there may be other trees which are not subtrees of T with error less than ϵ_{opt}, especially if T was constructed by a greedy top-down heuristic.) Thus, if we only compare our error to that of T_{opt}, we are effectively only paying a penalty of the same order that T_{opt} pays. If s_{opt} is small compared to m — that is, the optimal subtree of T is small — then this is quite good indeed.

But a stronger result is possible and desirable. Suppose that T_{opt} is not particularly small, but that there is a much smaller subtree T' whose error is not much worse than ϵ_{opt}. In such a case, we would rather claim that our error is close to that of T', with a penalty that goes only like $\sqrt{s'/m}$. This was the *index of resolvability* criterion for model selection first examined for density estimation by Barron and Cover [1], and we now generalize our main result to this setting.

Theorem 6 *Let S be a random sample of size m drawn according an unknown target function and input distribution. Let $T = T(S)$ be any decision tree, and let T^* denote the subtree of T output by our pruning algorithm on inputs S and T. Then with probability $1-\delta$ over S,*

$$\begin{aligned}\epsilon(T^*) \le \min_{T'} \Bigg\{ & \epsilon(T') + \\ & O\left(\left(\log(s_{\mathit{eff}}(T')/\delta) + \ell_{\mathit{eff}}(T')\sqrt{\log(n) + \log(m/\delta)}\right)\sqrt{\frac{s_{\mathit{eff}}(T')}{m}}\right)\Bigg\}. \end{aligned} \tag{30}$$

Here the min *is taken over all subtrees T' of T, and we define the "effective" size*

$$s_{\mathit{eff}}(T') = s' + 2m(\epsilon(T') - \epsilon_{opt}) + 6s' \log(s'/\delta) \tag{31}$$

and the 'effective" depth $\ell_{\mathit{eff}}(T') = \min\{\ell_{opt}, s'\}$, where s' and ℓ' are the size and depth of T', ϵ_{opt} denotes the smallest generalization error among all subtrees of T, and ℓ_{opt} denotes the depth of the subtree achieving ϵ_{opt}.

The proof is omitted due to space considerations, but the main difference from the proof of Theorem 5 is that our pruning is no longer a subtree of the pruning T' to which it is being compared. This requires a slight modification of the pruning penalty $\alpha(m_v, s_v, \ell_v, \delta)$, and the analysis bounding the sum of the sizes of the pruned subtrees becomes more involved.

Again ignoring logarithmic and depth factors for simplicity, Theorem 6 compares the error of our pruning *simultaneously* to *all* prunings T'. Our additional error goes roughly like $\sqrt{s_{\mathit{eff}}(T')/m}$. If s' is small compared to m and $\epsilon(T')$ is not much larger than ϵ_{opt}, then the bound shows that our error will compare well to ϵ_{opt} — even though the tree T' achieving the min may not be T_{opt}. This is the power of index of resolvability results.

References

[1] Andrew R. Barron and Thomas M. Cover. Minimum Complexity Density Estimation. *IEEE Transactions on Information Theory*, Vol. 37, No. 4, pages 1034 – 1054, 1991.

[2] Marco Bohanec and Ivan Bratko. Trading Accuracy for simplicity in Decision Trees. *Machine Learning*, Vol. 15, pages 223 – 250, 1994.

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

[4] David P. Helmbold and Robert E. Schapire. Predicting Nearly as Well as the Best Pruning of a Decision Tree. *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, ACM Press, pages 61 – 68, 1995.

[5] Michael Kearns and Yishay Mansour. On the Boosting Ability of Top-Down Decision Tree Learning Algorithms. *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, ACM Press, pages 459–468, 1996.

[6] M. Kearns, Y. Mansour, A. Ng, D. Ron. An Experimental and Theoretical Comparison of Model Selection Methods. *Machine Learning*, 27(1):7–50, 1997.

[7] Yishay Mansour. Pessimistic Decision Tree Pruning Based on Tree Size. *Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann, pages 195 – 201, 1997.

[8] J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[9] V.N. Vapnik and A. Ya. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to thier Probabilities. *Theory of Probability and its Applications*, XVI(2):264–280,1971.

[10] V.N. Vapnik. Estimation of Dependences Based on Empirical Data. Springer-Verlag, 1982.