Learning random log-depth decision trees under the uniform distribution

Jeffrey C. Jackson*
Department of Mathematics and Computer Science
Duquesne University, Pittsburgh, PA 15282
jackson@mathcs.duq.edu

Rocco A. Servedio†
Department of Computer Science
Columbia University, New York, NY 10027
rocco@deas.harvard.edu

November 30, 2004

Abstract

We consider three natural models of random logarithmic depth decision trees over Boolean variables. We give an efficient algorithm that for each of these models learns all but an inverse polynomial fraction of such trees using only uniformly distributed random examples from $\{0,1\}^n$. The learning algorithm constructs a decision tree as its hypothesis.

Keywords: PAC learning, decision trees

*This material is based upon work supported by the National Science Foundation under Grant No. CCR-0209064.
†Partially supported by NSF Early Career Development (CAREER) Grant CCF-0347282.

1 Introduction

Decision trees are widely used to represent various forms of knowledge. The apparent ease with which humans can understand and work with decision trees has also made them a popular form of representation for knowledge obtained through heuristic machine learning algorithms (see, e.g., [4, 9]). While heuristic algorithms have proved reasonably successful for many applications, there is some reason to believe that arbitrary decision trees are not efficiently learnable from random examples alone, as the class of decision trees is provably not efficiently learnable in the Statistical Query model, even when the examples are uniformly distributed [3].

Given the apparent difficulty of learning decision trees in polynomial time, many researchers have considered alternate learning scenarios. One line of work which has been pursued is to consider algorithms that run in superpolynomial time. Ehrenfeucht and Haussler [6] have shown that the class of size-s decision trees over $\{0,1\}^n$ can be PAC learned in $n^{\log s}$ time steps. This result was later simplified by Blum [1]. Another approach which has been pursued is to study decision tree learnability in alternate learning models in which the learner has more power and the classifier produced need not itself be a decision tree. Kushilevitz and Mansour [8] gave a polynomial time algorithm which uses membership queries and can learn decision trees under the uniform distribution. The hypothesis produced by this algorithm is a weighted threshold of parity functions. Using different techniques, Bshouty [5] gave a polynomial time algorithm which learns decision trees in the model of exact learning from membership and (non-proper) equivalence queries; this implies that decision trees can be PAC learned in polynomial time under any distribution if membership queries are allowed. The hypothesis in this case is a depth-three Boolean circuit.

1.1 Our approach and results

In this work we propose a third approach to coping with the difficulty of learning decision trees: looking at the average case. Since we have been unable to design algorithms which can learn all decision trees, we focus instead on algorithms which can efficiently learn most decision trees. Also, unlike some of the earlier approaches, our hypothesis is a decision tree.

We consider three natural models of random log-depth decision trees, i.e. decision trees over n Boolean variables which are of depth $O(\log n)$. (Note that decision trees of logarithmic depth are a natural class to study in the context of uniform distribution learning, since under the uniform distribution any decision tree of poly(n) size can be approximated to any inverse polynomial accuracy by a decision tree of $O(\log n)$ depth.) Our main result is a polynomial time algorithm that for each of these models learns, using decision tree hypotheses, all but a $1/p(n)$ fraction of such trees using only uniformly distributed random examples, where $p(n)$ is any polynomial function of n.

There are several motivations for this study. One natural motivation is that since decision tree learning is an interesting and important problem, but worst-case theoretical analyses seem quite hard, it is natural to consider the average case. Another motivation comes from work of Blum *et al.* [2], who proposed an approach to constructing cryptographic primitives such as pseudorandom generators and one-way functions based on the (presumed) intractability of certain learning problems. They defined a framework of learning from uniformly random examples in which the target function is also selected at random from the concept class according to some distribution. One of the main results of [2] is a proof that the existence of concept classes which are hard to learn in this average-case sense implies the existence of corresponding one-way functions whose circuit complexity is closely related to the circuit complexity of the concepts in the hard-to-learn class. Since decision trees are "simple" in terms of circuit complexity yet are widely viewed as an intractable learning problem, one natural application of the Blum *et al.* paradigm would be to construct one-way functions based on the presumed intractability of decision tree learning. However, our results (which show that log-depth decision trees are *not* hard to learn in the average case) indicate that this approach does not yield secure one-way functions.

In Section 2 we give the necessary background on uniform distribution learning and decision trees, and describe the three models of random decision trees which we consider. Section 3 gives useful Fourier properties of decision trees. In Section 4 we present the learning algorithm, and Sections 5 through 9 contain the proofs of correctness for the learning algorithm. We conclude in Section 10.

2 Preliminaries

A *Boolean decision tree* T is a rooted binary tree in which each internal node has two children and is labeled with a variable, and each leaf is labeled with a bit $b \in \{-1, +1\}$. Children are ordered, i.e., each internal node has a definite left child and right child. We refer to an internal node whose two children are both leaves as a *pre-leaf node*. Because we will deal exclusively with Boolean decision trees in this paper, for convenience we will refer to them simply as decision trees.

A decision tree T computes a Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$ in the obvious way: on input x, if variable x_i is at the root of T we go to either the left or right subtree depending on whether x_i is -1 or 1. We continue in this fashion until reaching a bit leaf; the value of this bit is $f(x)$.

We define the *depth of a node* in a decision tree as follows. First, every decision tree must have at least one node; we do not admit the empty (0-node) decision tree. In a tree consisting of a single leaf node (labeled with some bit), the depth of this node is -1; we call such a tree *trivial*. The depth of the root in a non-trivial tree is 0, and the depth of any non-root node is one greater than the depth of its parent. The *depth of a decision tree* T is -1 if T is trivial and the maximum depth over all pre-leaf nodes of T otherwise.

A decision tree is *non-redundant* if no variable occurs more than once on any root-to-leaf path. We consider only non-redundant decision trees in this paper.

We let $\mathcal{U}$ be the uniform distribution on $\{-1, 1\}^n$. We write $EX(f, \mathcal{U})$ to denote a uniform random example oracle for $f : \{-1, 1\}^n \to \{-1, 1\}$ which, when invoked, outputs a labeled example $\langle x, f(x)\rangle$ where x is drawn from $\mathcal{U}$.

We consider three models of random decision trees. Our primary model is the uniform distribution over the set of all non-redundant decision tree representations of depth at most d on the variable set $\{x_1, \ldots, x_n\}$. We call this the *uniform* model and will represent this distribution by $\mathcal{T}^U_{d,n}$. Note that not every Boolean function that can be represented by a depth-d tree has the same probability mass under $\mathcal{T}^U_{d,n}$; some functions may have more $\mathcal{T}^U_{d,n}$-good trees which represent them than others. That is, $\mathcal{T}^U_{d,n}$ is a distribution over syntactic representations of decision tree functions, and not over the functions themselves.

In each of our other two models, the internal nodes form a complete tree of depth d and are labeled uniformly at random using the variables $\{x_1, \ldots, x_n\}$, with the restriction that the tree must be non-redundant. These models, denoted by $\mathcal{T}^C_{d,n}$ and $\mathcal{T}^B_{d,n}$, differ in that the leaves in $\mathcal{T}^C_{d,n}$ are selected independently and uniformly from $\{-1, 1\}$ while in $\mathcal{T}^B_{d,n}$ each sibling pair must have opposite signs, although the sign of the left node is independently and uniformly chosen from $\{-1, 1\}$. We call these the *complete* and *balanced* models, respectively.

We assume throughout that d is $O(\log n)$, and that the learning algorithm knows the exact value of d. This latter assumption is without loss of generality since the algorithm can try all values $d = 1, 2, \ldots$ until it succeeds.

3 Fourier properties of decision trees

We will be interested in carefully measuring the correlation between a decision tree f and each of f's variables. Define e_i to be the n-bit vector that has a 1 in position i and 0's elsewhere and define $\hat{f}(e_i)$ to be $E_{a \sim \mathcal{U}}[a_i f(a)]$. Since $f(a)$ and a_i take values in $\{-1, 1\}$ we have $\hat{f}(e_i) = \Pr_{a \sim \mathcal{U}}[f(a) = a_i] - \Pr_{a \sim \mathcal{U}}[f(a) \neq a_i]$. Each $\hat{f}(e_i)$ is a *first-order Fourier coefficient* of f.

Kushilevitz and Mansour [8] showed that decision trees have some particularly useful Fourier properties.[1] Define $L(i)$ to be the set of all leaves in a decision tree f that are descendants of some node labeled by variable x_i, and let $d(\ell)$ represent the depth of a leaf ℓ in f. The analysis of [8] directly implies:

Corollary 1 (Kushilevitz Mansour) *For every decision tree f and every $1 \leq i \leq n$, there is a function $\sigma_i : L(i) \rightarrow \{-1, 1\}$ such that*

$$\hat{f}(e_i) = \sum_{\ell \in L(i)} 2^{-d(\ell)} \sigma_i(\ell).$$

Note that this corollary implies that in any tree of depth d, each $\hat{f}(e_i)$ is of the form $j/2^d$ for some integer j. This is because any leaf at depth $d+1$ must have a sibling leaf, so the total number of $\pm 1/2^{d+1}$ contributions to $\hat{f}(e_i)$ is even. We say that any first-order Fourier coefficient of the form $\frac{2k+1}{2^d}$ for some integer k is an *odd coefficient*, and all other first-order coefficients are *even coefficients*.

From the above corollary we can obtain the following:

Lemma 2 *For every decision tree f of depth d and every $1 \leq i \leq n$, $\hat{f}(e_i)$ is an odd coefficient if and only if the total number of occurrences of the following conditions is odd for x_i:*

1. *A node at depth d is labeled with x_i and the children of this node (both leaves) have opposite signs;*

2. *A leaf at depth d has an ancestor labeled with x_i;*

3. *A pair of sibling leaves at depth $d+1$ have the same sign, x_i labels an ancestor of this pair of leaves, and x_i is not the label of the parent of the pair.*

Proof: Fix an arbitrary i. By the corollary, only leaves in $L(i)$ that are at depth d or $d+1$ can affect whether $\hat{f}(e_i)$ is an even or odd coefficient. Also, as noted earlier, each leaf at depth $d+1$ has a sibling leaf. Let "leaf set" denote either a leaf at depth d or a sibling pair of leaves at depth $d+1$. Then notice that every leaf set in $L(i)$ is covered by at most one of the conditions of the lemma. We will show that each leaf set in $L(i)$ that is not covered by a condition contributes nothing to $\hat{f}(e_i)$ while every covered leaf set contributes $\pm 1/2^d$. From this, the lemma follows.

First, again by the corollary, each leaf in $L(i)$ at depth d contributes $\pm 1/2^d$ to $\hat{f}(e_i)$. For sibling leaves at depth $d+1$ there are two cases: the siblings either have the same sign or different signs. If sibling leaves at depth $d+1$ have the same sign, then their immediate parent is irrelevant: the tree is equivalent to one with a leaf at depth d in place of the parent node. Thus, in this case, the non-parent ancestor nodes each receive a $\pm 1/2^d$ contribution to their corresponding Fourier coefficients while the parent node receives no contribution to its coefficient. For the case in which the siblings have different signs, it is not hard to see from the definition of $\hat{f}(e_i)$ that the net contribution to

[1] [8] considered decision trees in which internal nodes can contain arbitrary parity functions; however as noted earlier we only allow single variables at internal nodes.

$\hat{f}(e_i)$ of the set of all bit-vectors a that reach these two leaves will be $\pm 1/2^d$ if x_i is the parent of the leaves and 0 if x_i is a non-parent ancestor of the leaves. ■

Conditions 2 and 3 of this lemma may seem redundant, since a tree with two sibling leaves having the same sign is equivalent to a tree that has a single leaf in place of the parent of the siblings. We include both conditions because these are syntactically different structures both of which may arise in the various trees we consider.

A key observation which follows from this lemma is:

Lemma 3 *Fix any Boolean decision tree structure of depth d and assign variables x_1 through x_n arbitrarily to the internal nodes of the tree, with the constraint that the resulting tree is non-redundant. Assign each leaf bit by independently and uniformly selecting from $\{-1, 1\}$. Then for every $1 \leq i \leq n$ such that x_i is an ancestor of at least one leaf at depth $d+1$, we have* $\Pr[\hat{f}(e_i)$ *is an odd coefficient* $] = \frac{1}{2}$.

Moreover, let S be any subset of variables $x_1, \ldots, x_n$ with the following property: there is a collection C of $|S|$ pre-leaves in T, each of which is at depth d and is labeled with a different element of S, such that no variable in S occurs on any of the paths from the root to any of these pre-leaves. Then we have that $\Pr[\forall i \in S\ \hat{f}(e_i)$ *is an even coefficient* $] = \frac{1}{2^{|S|}}$.

Proof: We can view certain leaves of the tree as defining the "parity" of the internal nodes in a way that corresponds to the conditions of Lemma 2. More precisely, all internal nodes begin with even parity, and then their parities are computed by applying the following rules to each leaf and each pair of leaves (each leaf or leaf pair will meet the conditions of at most one rule):

1. If a pair of sibling leaves are at depth $d+1$ and have opposite signs, then the parity of their depth-d parent node is toggled.

2. If a leaf is at depth d then the parity of each ancestor node is toggled.

3. If a pair of sibling leaves are at depth $d+1$ and have identical signs, then the parity of each ancestor node except their parent node is toggled.

We can then define the parity of a variable x_i as the parity of the parity of all nodes that are labeled by x_i. It is clear that for each i, the first-order Fourier coefficient $\hat{f}(e_i)$ is odd if and only if x_i has odd parity according to this definition.

Next, notice that since sibling leaves are labeled uniformly at random, any pair of sibling leaves at depth $d+1$ is equally likely to satisfy rule 1 or rule 3 above. So the probability that any given ancestor node of a pair of such sibling leaves has its parity toggled by a random assignment to these leaf bits is exactly $1/2$. The same is true of the parity of the variable labeling the node. Since all leaves at depth $d+1$ have sibling leaves (because the tree depth is d), all possible assignments to leaves at depth $d+1$ are covered by the conditions of rules 1 and 3. Thus, any variable x_i that is an ancestor of at least one leaf at depth $d+1$ will have odd parity with probability exactly $1/2$, and the first equation is proved.

To see that the second equation also holds, observe that the $|S|$ pre-leaves in C will each toggle the parity of the corresponding variable with probability $1/2$. Since no variable in S occurs on any path to a node in C, the final parities of these variables are all independent of each other, and the lemma follows. ■

Kushilevitz and Mansour also observed that it follows from Corollary 1 and Chernoff bounds that for any decision tree f and any $\delta > 0$, a uniformly distributed sample S of m labeled pairs $\langle x, f(x) \rangle$ is—with probability at least $1 - \delta$ over the choice of S—sufficient to compute all of the first-order Fourier coefficients of f exactly, for m exponential in the depth d of f and polynomial in $\log(n/\delta)$. In particular, with probability at least $1-\delta$ over the random draw of S, $\hat{f}(e_i) = R\left((\sum_{a \in S} a_i f(a))/m\right)$ where $R(\cdot)$ represents "rounding" the argument to the nearest rational number having denominator 2^d. Therefore, we also have

Corollary 4 (Kushilevitz Mansour) *There is an algorithm* **FCExact** *such that, given* $\delta > 0$ *and access to* $EX(f,\mathcal{U})$ *for any decision tree* f *of depth* $O(\log n)$*, with probability at least* $1-\delta$ **FCExact**$(n, \delta, EX(f,\mathcal{U}))$ *computes all of the first-order Fourier coefficients of* f *exactly in time poly*$(n, \log(1/\delta))$.

For our algorithm we will need uniform random examples which are labeled not only according to the original tree f, but also according to certain subtrees obtained by restricting a subset of the variables of f. Each such subset will lie along a root-to-leaf path in f and—since we consider only trees of depth $O(\log n)$—will therefore have cardinality $O(\log n)$. We can simulate exactly an example oracle for such a restricted f' given an oracle $EX(f,\mathcal{U})$ by simply drawing examples from $EX(f,\mathcal{U})$ until we obtain one that satisfies the restriction on the $O(\log n)$ variables. Since each example from $EX(f,\mathcal{U})$ will satisfy such a restriction with probability $1/\text{poly}(n)$, the probability of failing to obtain such an example after poly(n) many draws from $EX(f,\mathcal{U})$ can be made exponentially small. Thus the simulation of these subtree oracles is both exact and efficient. We will use $EX(f',\mathcal{U})$ to represent the simulated oracle for a restriction f'.

4 The algorithm for learning random decision trees

Our algorithm for learning random decision trees, which we call **LearnTree**, operates in two phases. In the first phase (lines 4-11) the algorithm uses the Fourier properties outlined above to find a root-like variable for the original tree. (Informally, a root-like variable has the property that it can be taken as the root of the tree without increasing the depth of the tree; we give precise definitions later.) Once this is done, it recursively finds a good root for each of the two subtrees induced by this root, and so on. The process stops at depth $d - \frac{1}{2}\log n$, so when it stops there are at most $2^d/\sqrt{n}$ subtrees remaining, each of depth at most $\frac{1}{2}\log n$. (Recall that w.l.o.g. we may assume the algorithm knows the exact value of d.) In the second phase (lines 1-3) we employ an algorithm **UnikDTLearn**$(n, \epsilon, \delta, EX(T,\mathcal{U}))$ due to Hancock [7] to learn the remaining "shallow" decision trees. (If $d < \frac{1}{2}\log n$ then our algorithm only performs the second phase.)

The intuition underlying the algorithm is that at each step in the first phase, each of the two subtrees of the root x_i of a decision tree T will obviously have depth at least one less than that of the original tree. These subtrees will therefore contain no odd first-order Fourier coefficients by Lemma 2, and thus the root x_i will pass the test at line 8. On the other hand, we will show that in our random decision tree models, if we consider a variable x_j which is not the root (or, more accurately, is not root-like in a sense defined below) then projecting on x_j will result in at least one projection containing odd first-order coefficients. (This will follow from our earlier Fourier analysis of decision trees plus some combinatorial arguments showing that if x_j is not root-like, then with very high probability the trees resulting from restricting on x_j will have an $\omega(\log n)$ size collection C of pre-leaves as in Lemma 3.)

Hancock's **UnikDTLearn** is efficient, so **LearnTree** clearly runs in time poly(n) for any value $d = O(\log n)$. The rest of this paper shows that the algorithm with high probability outputs an accurate decision tree for our various models of random decision trees.

LearnTree$(n, d, \delta, \epsilon, EX(T,\mathcal{U}))$
(is given number of variables n, the depth d of the tree T to be learned, desired confidence $\delta > 0$, desired accuracy $\epsilon > 0$, and access to $EX(T,\mathcal{U})$)

1. **if** $d \leq (\frac{1}{2}\log n)$
2.   **return UnikDTLearn**$(n, \epsilon, \delta, EX(T,\mathcal{U}))$
3. **endif**
4. **for** $i = 1 \ldots n$
5.   **for** $b = -1, 1$
6.   Call **FCExact**$(n-1, \delta/(4n^2), EX(T_{x_i \leftarrow b},\mathcal{U}))$ to compute $\widehat{T_{x_i \leftarrow b}}(e_j)$ for all $j \neq i$
7.   **endfor**
8.   **if** none of the coefficients $\widehat{T_{x_i \leftarrow b}}(e_j)$ is odd
9.   **return** tree consisting of root x_i and children defined by **LearnTree**$(n-1, d-1, \delta/4, \epsilon, EX(T_{x_i \leftarrow b}),\mathcal{U})$ for $b = -1, 1$.
10.   **endif**
11. **endfor**
12. **return** "fail"

Figure 1: The algorithm for learning random log-depth decision trees.

5 Bottlenecks and recursing the algorithm

The first phase of our algorithm attempts to recursively select the root of the original tree T and its subtrees. One difficulty is that T may have more than one variable that "acts like" a root, in the sense that putting any of these variables at the root does not increase the depth of the tree. For example, consider a decision tree representation of the function $x_1 \oplus x_2$. Although we might represent this with a tree T having x_1 at the root, it can be represented equally well by a tree with x_2 at the root.

The following definition captures the notion of a "root-like" variable for us:

Definition 5 *A variable x_i is a* bottleneck *for a decision tree T if T is non-trivial and x_i occurs on every root-to-leaf path in T.*

Clearly the variable labeling the root is always a bottleneck for any tree. We note that if x_i is the root of a tree T, then a variable $x_j \neq x_i$ is a bottleneck in T if and only if x_j is a bottleneck in both the left and right subtrees of T. Notice also that any bottleneck variable will pass the test at line 8 of **LearnTree**, so some variable x_i will pass the test at every stage of the recursion given that **FCExact** returns accurate values for all first-order Fourier coefficients.

For the rest of this section let $\mathcal{T}_{d,n}$ denote any fixed one of the three random tree models $\mathcal{T}^B_{d,n}, \mathcal{T}^C_{d,n}, \mathcal{T}^U_{d,n}$. In later sections we will show that for a random tree T drawn from $\mathcal{T}_{d,n}$, any non-bottleneck variable will with very high probability not pass the test of line 8. Thus each recursive call of **LearnTree** is performed by restricting on some bottleneck variable; however, the bottleneck may or may not be the root. If the root is the bottleneck chosen, then it is easy to see that each of the two subtrees will be drawn from $\mathcal{T}_{d-1,n-1}$ (over a suitable set of $n-1$ variables) as desired, and the inductive assumption of **LearnTree** (that the tree it is given is drawn from $\mathcal{T}_{d,n}$) will be valid. If a non-root bottleneck is chosen, however, it is not *a priori* clear that the two resulting subfunctions for the recursive call correspond to draws from $\mathcal{T}_{d-1,n-1}$.

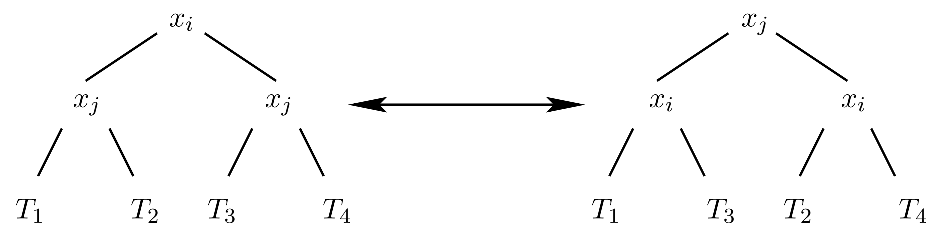

Figure 2: Applying a root swap operation to the tree on the left (right) produces the tree on the right (left). Note that the tree produced by a root swap computes the same function as the original tree.

We will show shortly that as long as any bottleneck is chosen, the two resulting subfunctions do indeed correspond to draws from $\mathcal{T}_{d-1,n-1}$. (The correspondence is exact for the complete model; for the other models the subfunctions correspond to draws from a distribution which has negligible statistical difference from $\mathcal{T}_{d-1,n-1}$.) While the two draws are not independent of each other, this does not negatively impact the algorithm.

First, we define some terms. Figure 2 defines a tree restructuring operation we call a *root swap*. Notice that this operation can be performed on a tree only if the children of the root are both labeled with the same variable; we call a tree with this property *root swappable*. More generally, a *swap* operation takes a tree T and a node η in the tree such that the subtree S rooted at η is root swappable, and returns a tree T' which is identical to T except that the subtree rooted at η is replaced with the root swap of S.

We next define an equivalence relation on decision trees which we call *structural equivalence*. Formally, this relation is defined inductively as follows:

Definition 6 *(i) Two decision trees T and T' both of depth $d < 1$ are structurally equivalent if and only if they are identical. (ii) Two decision trees T and T' both of depth $d \geq 1$ are structurally equivalent if and only if there exists a sequence of swap operations that will transform T' to a tree T'' such that T and T'' have the same root variable and each of the child subtrees of the root of T'' is structurally equivalent to its corresponding subtree in T. (iii) Two decision trees T and T' of different depths are not structurally equivalent.*

Informally, two decision trees T and T' are structurally equivalent if T' can be obtained from T by performing a sequence of swaps. Note that if two trees are structurally equivalent then they compute the same function. The following lemma, which we prove in Appendix A, shows that bottleneck variables can be swapped to the root of a tree in a way that preserves structural equivalence.

Lemma 7 *Let T be any decision tree. If variable x_i is a bottleneck for T, then there is a tree T' having x_i at its root that is structurally equivalent to T.*

Let $\mathcal{T}^i_{d,n}$ be the induced distribution over trees obtained by restricting $\mathcal{T}^C_{d,n}$ to trees for which x_i is a bottleneck, and let $\mathcal{T}^{\tilde{i}}_{d,n}$ be the distribution over trees obtained by first selecting a tree T according to $\mathcal{T}^i_{d,n}$ and then performing a minimal sequence of swap operations (as implicitly described in the proof Lemma 7) to produce a structurally equivalent $\tilde{T}$ having x_i as its root. Finally, let $\mathcal{T}^{-1}_{d-1,n-1}$ (resp. $\mathcal{T}^{1}_{d-1,n-1}$) represent the distribution over trees corresponding to a random variable that selects a tree $\tilde{T}$ according to $\mathcal{T}^{\tilde{i}}_{d,n}$ and then returns the left (resp. right) subtree as the value of the random variable. Then for the complete model, it suffices to prove the following lemma:

Lemma 8 *Let $\mathcal{T}^i_{d,n}$, $\mathcal{T}^{-1}_{d-1,n-1}$ and $\mathcal{T}^1_{d-1,n-1}$ be as defined above. Then for all $1 \le i, d \le n$, $\mathcal{T}^{-1}_{d-1,n-1}$ and $\mathcal{T}^1_{d-1,n-1}$ are both identical to $\mathcal{T}^C_{d-1,n-1}$.*

Proof: The proof is by induction on d. For the base case $d = 1$, any tree T drawn from $\mathcal{T}^i_{1,n}$ either has x_i at the root or some other variable x_j at the root and x_i as the root of both children of x_j. In either case, the corresponding tree $\tilde{T}$ of the process defining $\mathcal{T}^{\tilde{i}}_{1,n}$ will have x_i at the root with two depth 0 children. It is easy to see that, over random draws from $\mathcal{T}^{\tilde{i}}_{1,n}$, the root variables of these children of x_i are each uniformly distributed over the $n-1$ variables excluding x_i (although the distributions of these root variables are not necessarily independent). The values of the leaves are also uniformly and independently distributed. Therefore, the base case has been shown.

For the inductive case, consider a tree T drawn from $\mathcal{T}^i_{d,n}$, for fixed $d > 1$. Since x_i must be a bottleneck in T, it is either the root of T or is a bottleneck in both children of the root of T. If x_i is the root of T, the lemma obviously holds. So we are left with the case in which some variable x_j—uniformly chosen from the $n-1$ variables excluding x_i—labels the root of T and x_i is a bottleneck in both children of x_j. Let T_{-1} (T_1) represent the left (right) subtree of T, and let $\tilde{T}_{-1}$ ($\tilde{T}_1$) represent the tree obtained by swapping x_i to the root of T_{-1} (T_1). Since x_i is a bottleneck in T_{-1} (T_1), the children of x_i in $\tilde{T}_{-1}$ ($\tilde{T}_1$) are drawn from $\mathcal{T}^C_{d-2,n-2}$, by the inductive hypothesis.[2] Notice also that, although the distribution over each child of x_i in $\tilde{T}_{-1}$ may be dependent on its sibling's distribution, each is independent of both of the distributions over children of x_i in $\tilde{T}_1$. Therefore, after performing a final swap of the x_i's at the roots of $\tilde{T}_{-1}$ and $\tilde{T}_1$ with the root x_j of T, we obtain a tree $\tilde{T}$ in which each child of the root x_i is a tree rooted at uniformly (over $n-1$ variables) chosen x_j and in which the children of x_j are independently distributed according to $\mathcal{T}^C_{d-2,n-2}$. That is, each child of x_i in $\tilde{T}$ is distributed according to $\mathcal{T}^C_{d-1,n-1}$. Since $\tilde{T}$ is by construction distributed according to $\mathcal{T}^{\tilde{i}}_{d,n}$, the lemma follows. ■

This proof does not immediately apply to either of the other two tree models. The problem for balanced trees is in the base case: a swap in a balanced tree of depth 1 can produce a tree that is no longer balanced. For uniform trees, there is a technical difficulty in that the distribution over children will only be exactly $\mathcal{T}^U_{d-1,n-1}$ in the case in which no swap is performed (the root is chosen as the bottleneck variable). If a non-root bottleneck is chosen, a swap must be performed, and the children of the root then must not be leaves. But leaves have nonzero probability in $\mathcal{T}^U_{d-1,n-1}$, so the children are not distributed according to $\mathcal{T}^U_{d-1,n-1}$ in this case.

We can, however, still say that with very high probability over the choice of tree according to either $\mathcal{T}^B_{d,n}$ or $\mathcal{T}^U_{d,n}$, the choice of bottleneck variable on which to recurse will not affect the distribution over the recursive subtree problems. We use two lemmas to show this. First, we will later prove the following lemma in the context of some of our other combinatorial lemmas in Appendix E.

Lemma 9 *Let T be drawn from $\mathcal{T}_{d,n}$ where $d \ge \frac{1}{2}\log n$ and $d = O(\log n)$. The probability that T has any bottleneck variable which occurs at some depth $k \ge \frac{1}{8}\log n$ is $1/n^{\omega(1)}$.*

For the balanced model, this means that the sequence of swaps performed to transform T chosen according to $\mathcal{T}^B_{d,n}$ into $\tilde{T}$ having bottleneck x_i at the root will almost certainly not perform a swap involving x_i nodes whose children are leaf bits. And a simple modification to the proof of Lemma 8 shows the following:

[2] Strictly speaking, the children of x_i are fixed for any given T. What we are actually claiming here is that over draws of T from $\mathcal{T}^i_{d,n}$, for fixed $d > 1$, the children of x_i in $\tilde{T}_{-1}$ are distributed according to $\mathcal{T}^C_{d-2,n-2}$. But for ease of exposition, here and below we often blur the distinction between a single tree produced by one application of a random process defining a distribution over trees and the distribution itself.

Lemma 10 *Let $\mathcal{T}^{\bar{i}}_{d,n}$ be the induced distribution over trees obtained by restricting $\mathcal{T}^{B}_{d,n}$ to trees for which x_i is a bottleneck that does not appear at depth d. Let $\mathcal{T}^{-1}_{d-1,n-1}$ and $\mathcal{T}^{1}_{d-1,n-1}$ be induced distributions defined as before except relative to $\mathcal{T}^{\bar{i}}_{d,n}$ rather than $\mathcal{T}^{i}_{d,n}$. Then for all $1 \leq i, d \leq n$, $\mathcal{T}^{-1}_{d-1,n-1}$ and $\mathcal{T}^{1}_{d-1,n-1}$ are both identical to $\mathcal{T}^{B}_{d-1,n-1}$.*

So in the balanced model, there is negligible chance that the choice of bottleneck variable will negatively affect the algorithm. For the uniform model, we will also use the following lemma, which is again proved in Appendix E.

Lemma 11 *Let T be drawn from $\mathcal{T}^{U}_{d,n}$ where $d \geq \frac{1}{4}\log n$. The probability that T has a leaf at depth less than $d - \frac{1}{4}\log n$ is $1/2^{n^{\Omega(1)}}$.*

Combining this with Lemma 9, for $d \geq \frac{1}{2}\log n$, with probability at least $1 - 1/n^{\omega(1)}$ (which we will refer to in this section as "very high probability"), a tree T drawn from $\mathcal{T}^{U}_{d,n}$ has no leaves through depth $d - \frac{1}{4}\log n$, while all bottleneck variables are at shallower depths. Fix t to be any depth ($\frac{3}{16}\log n$ will do) such that with very high probability all the bottleneck variables in T drawn from $\mathcal{T}^{U}_{d,n}$ have depth less than t and all leaves have depth greater than t.

Notice that (with very high probability) we have 2^t nodes at depth t in T and that the subtrees rooted at these nodes are independently and identically (up to variable renaming) distributed according to $\mathcal{T}^{U}_{d-t,n-t}$. Furthermore, notice that all of the nodes labeled by any bottleneck variable are with very high probability located in the portion of the tree above these new "leaves". So the proof of Lemma 8 shows that, with very high probability, swapping a bottleneck variable x_i to the root of T produces a tree $\tilde{T}$ in which the children of x_i are each distributed according to the complete distribution down to depth $t-1$ and then according to $\mathcal{T}^{U}_{d-t,n-t}$. But this is just $\mathcal{T}^{U}_{d-1,n-1}$ restricted to contain no leaves until depth t, which by Lemma 11 is negligibly different from $\mathcal{T}^{U}_{d-1,n-1}$.

Therefore, in the uniform model, the recursive distributions obtained if a non-root bottleneck is chosen are negligibly different from the distributions that would be obtained if the root variable was selected. Thus, in both the balanced and the uniform models, the probability that **LearnTree** fails as a result of distribution differences induced by selecting a non-root bottleneck is negligibly small. This failure probability will be covered by a portion of the PAC confidence parameter δ.

It remains to show that **LearnTree** will with high probability choose a bottleneck at each stage of the recursion in the first phase, and that Hancock's algorithm can be used to efficiently learn $\mathcal{T}_{d,n}$-random trees of depth $\frac{1}{2}\log n$ with high probability. We address the second point first in the next section.

6 Learning random $(\frac{1}{2}\log n)$-depth trees

We stop the recursion in **LearnTree** at depth $\frac{1}{2}\log n$ because our analysis depends on trees being somewhat deep. So we use another method for learning random trees of depth less than $\frac{1}{2}\log n$, which is based on the following lemma plus the **UnikDTLearn** algorithm due to Hancock [7].

Recall that a decision tree T is *read-k* if each variable labels at most k nodes in T. We again write $\mathcal{T}_{d,n}$ to represent any of our three random tree models. We prove the following easy lemma in Appendix B:

Lemma 12 *Let $r = ((1-\epsilon)\log n) - 2$ for some constant $\epsilon > 0$. Let C be any constant. Then we have $\Pr_{T \in \mathcal{T}_{r,n}}[T$ is not read-$k] \leq 1/n^C$ for $k = (C+2)/\epsilon$.*

Thus, if $r = (\frac{1}{2}\log n)$, then for any constant C we may take $k = 8C + 16$, and with probability at least $1 - \frac{4}{(n-2)^C}$ a tree T drawn from $\mathcal{T}_{d,n}$ is read-k (since each of the four subtrees of depth $r-2$ is not read-$(2C+4)$ with probability at most $1/(n-2)^C$).

Hancock [7] has given an algorithm **UnikDTLearn** and shown that it (or more precisely, a version which takes k as an input along with the parameters given earlier) efficiently learns read-k trees with respect to the uniform distribution, producing a decision tree (not necessarily read-k) as its hypothesis. Given any constant k, his algorithm terminates in time polynomial in n, $1/\epsilon$, and $1/\delta$, regardless of whether or not the target function f is actually a read-k decision tree. So our version of **UnikDTLearn**$(n, \epsilon, \delta, EX(T, \mathcal{U}))$ will begin by finding the smallest integer C such that $1/n^C \leq \delta/2$. If the δ originally provided to **LearnTree** is inverse polynomial in n, then this value C will be a constant independent of n. Taking $k = 8C + 16$, this means that the target function provided to **UnikDTLearn** is a read-k decision tree with probability at least $1-\delta/2$. Then running Hancock's original **UnikDTLearn** with this value of k and with $\delta/2$ as the confidence parameter will succeed at learning an ϵ-approximating tree with probability at least $1 - \delta/2$, for an overall success probability at the bottom of the recursion of $1 - \delta$. In short, we have the following:

Lemma 13 *If the function f in the oracle $EX(f, \mathcal{U})$ in the call to* **UnikDTLearn** *in* **LearnTree** *is distributed according to $\mathcal{T}_{d,n}$, then* **UnikDTLearn** *returns a decision tree that ϵ-approximates f with probability (over the random choice of T and the randomness in $EX(f, \mathcal{U})$) at least $1 - \delta$.*

It remains to show that the first stage of the algorithm successfully finds a bottleneck variable with high probability given a decision tree drawn at random according to one of our tree models and with depth at least $\frac{1}{2}\log n$. Throughout the rest of the paper we thus have $d = \Theta(\log n)$, $d \geq \frac{1}{2}\log n$. We will consider each model separately, beginning with the complete model.

7 Identifying bottlenecks in the complete model $\mathcal{T}^C_{d,n}$

Since we have already shown that any bottleneck makes an equally good root in the hypothesis, and since it is easily seen that all bottlenecks (including the root of T) will pass the test at line 8 of **LearnTree**, it remains to show the following: for each $i = 1, \ldots, n$, if x_i is *not* a bottleneck in a random tree T then the probability that x_i passes the test in line 8 is negligibly small.

Our general plan of attack is as follows: we will prove that if x_1 is *not* a bottleneck, then with $1 - \frac{1}{n^{\omega(1)}}$ probability there are many root-to-leaf paths in T that do not include x_1. We then argue that, conditioned on there being many such paths, among these pre-leaves there is a collection C satisfying the condition of Lemma 3 which has $|C| = \omega(\log n)$. Combining this with the Fourier properties of random decision trees derived earlier gives us our result.

More precisely, the argument is as follows. Let S be a random variable which denotes the number of x_1-free paths from the root to a pre-leaf in $T \in \mathcal{T}^C_{d,n}$. (Note that each such path ends at a depth-d pre-leaf since we are in the complete model. Note also that since we are in the complete model, $S > 0$ iff x_1 is not a bottleneck). We will prove the following lemmas in Appendix C.

Lemma 14 *For $0 \leq d \leq n-1$ we have* $\Pr_{T \in \mathcal{T}^C_{d,n}}[S = 0] \leq \frac{1}{n-d}$.

Lemma 15 *For any value $1 \leq k \leq (\log n)^{3/2}$ we have* $\Pr[S = k] = 1/n^{\omega(1)}$.

Lemma 16 *Let T be drawn from $\mathcal{T}^C_{d,n}$ conditioned on its having some set of $(\log n)^{3/2}$ pre-leaves at depth d, each of which has no x_1-labeled node as an ancestor. Then with probability $1 - 1/n^{\omega(1)}$ there is a set C of $(\log n)^{5/4}$ pre-leaves at depth d, each labeled with a distinct variable, each of which has no ancestor labeled with x_1 or with a variable that labels any element of C.*

From these lemmas it is easy to prove that each non-bottleneck will pass the test at line 8 with negligible probability:

Theorem 17 *Let $T \in \mathcal{T}_{d,n}^C$ where $d = \Theta(\log n), d \geq \frac{1}{2}\log n$. If x_1 is not a bottleneck then the probability that x_1 passes the test in line 8 is $1/n^{\omega(1)}$.*

Proof: Since $S = 0$ iff x_1 is a bottleneck, we have

$$\begin{aligned} \Pr_{T\in\mathcal{T}_{d,n}^C}[S < (\log n)^{3/2} \mid x_1 \text{ is not a bottleneck}] &= \frac{\Pr[S < (\log n)^{3/2} \ \&\ S > 0]}{\Pr[S > 0]} \\ &= \frac{\Pr[1 \leq S < (\log n)^{3/2}]}{\Pr[S > 0]} \\ &= 1/n^{\omega(1)} \end{aligned}$$

where the last equality follows from Lemmas 14 and 15. Thus we may assume that $S \geq (\log n)^{3/2}$. Lemma 16 now implies that there is a set C of $(\log n)^{5/4}$ pre-leaves with the stated properties. Now we observe that if a pre-leaf belongs to C, then under any restriction $x_1 \leftarrow b$, the pre-leaf will still occur at depth d with the desired property (that no variable labeling any node of C occurs as an ancestor of any node of C) in the tree resulting from the restriction. Thus by Lemma 3, the probability that all variables labeling nodes in C have even coefficients in the restricted tree is at most $1/2^{(\log n)^{5/4}} = 1/n^{\omega(1)}$. Hence x_1 passes the test at line 8 with negligible probability and the theorem is proved. ■

Combined with our earlier remarks, this establishes

Theorem 18 *For any n, any polynomial $p(\cdot)$, any $\delta > 1/p(n)$, and any $\epsilon > 0$, algorithm* **LearnTree** *will with probability at least $1 - \delta$ (over a random choice of tree T from $\mathcal{T}_{d,n}^C$ and the randomness of the example oracle) produce a hypothesis decision tree T' that ϵ-approximates the target with respect to the uniform distribution.* **LearnTree** *runs in time polynomial in n and $1/\epsilon$.*

Proof: By Lemma 13, the base case of the algorithm will succeed with probability at least $1 - \delta$ as long as it is run on a tree drawn from $\mathcal{T}_{c,n}^C$ for some $c \leq \frac{1}{2}\log n$. In the recursive phase, all first-order Fourier coefficients will be computed exactly with probability at least $1 - \delta/4$. Furthermore, assuming that the coefficients are correctly computed, every bottleneck variable will pass the test at line 8 of **LearnTree**, and by the preceding theorem the probability is negligible that any non-bottleneck variable will pass this test. Thus, in the recursive phase of the algorithm, with probability at least $1 - \delta/4$ a bottleneck variable will be chosen by the test. By the arguments of Section 5, the two functions obtained by restricting on either value of this bottleneck variable will both be distributed according to $\mathcal{T}_{d,n}^C$. Therefore, the two recursive calls to **LearnTree** will succeed with probability at least $1 - \delta/2$, so that overall the recursive phase succeeds with probability at least $1 - \delta$. Furthermore, it is easy to see that the tree returned by the recursive phase will be an ϵ-approximator to the target if each of the subtrees returned by the recursive call is an ϵ-approximator. Finally, for $d = O(\log n)$, the number of recursive calls is clearly polynomially bounded, and thus the algorithm runs in the time claimed given the previously mentioned bounds on **UnikDTLearn** and **FCExact**. ■

8 Identifying bottlenecks in the balanced model $\mathcal{T}^B_{d,n}$

We first need an analogue of Lemma 3 for the balanced model. Let T be a decision tree drawn from $\mathcal{T}^B_{d,n}$. Let T' be the tree that results from applying the restriction $x_1 \leftarrow 1$ to T, i.e. T' is obtained from T by replacing each occurrence of x_1 in T with its right subtree.

As in Lemma 3, each internal node of T' has a "parity" which is defined exactly as in the proof of Lemma 3, and each variable x_i has a parity which equals the parity of the parity of all nodes labeled by x_i in T'. It is easily seen that rule 1 contributes to the parity of x_i in T' precisely once for each occurrence of x_i as a pre-leaf in T which lies on an x_1-free path from the root in T; let $N_{i,1}$ denote the number of such occurrences. Rule 2 contributes to the parity of x_i in T' precisely once for each occurrence of x_1 as a pre-leaf in T which lies on a path from the root in T which contains x_i; let $N_{i,2}$ denote the number of such occurrences. (Rule 3 can never be satisfied since each pair of sibling leaf bits in T' must have opposite signs.) Note that the values of $N_{i,1}$ and $N_{i,2}$ are independent of whether we defined T' by the restriction $x_1 \leftarrow -1$ or $x_1 \leftarrow 1$ in the previous paragraph.

As in Section 7, to prove that **LearnTree** succeeds for random T drawn from $\mathcal{T}^B_{d,n}$, we must show that if x_1 is not a bottleneck in a random T drawn from $\mathcal{T}^B_{d,n}$, then the probability that x_1 passes the test in line 8 is $1/n^{\omega(1)}$. From the discussion of the previous paragraph, this is equivalent to showing that

$$\Pr[N_{i,1} + N_{i,2} \text{ is even for all } i = 2, \ldots, n] = 1/n^{\omega(1)}. \tag{1}$$

We prove this as follows. As in Section 7 let S be a random variable which denotes the number of x_1-free paths from the root to a pre-leaf in T. (The difference is that now T is drawn from $\mathcal{T}^B_{d,n}$ instead of $\mathcal{T}^C_{d,n}$; however these two distributions are identical as regards the internal nodes of a tree drawn from one of them.) By Lemmas 14 and 15 we know that conditioned on x_1 not being a bottleneck in T, we have $\Pr_{T \in \mathcal{T}^B_{d,n}}[S > (\log n)^{3/2}] = 1 - 1/n^{\omega(1)}$ (these lemmas are easily seen to hold for $\mathcal{T}^B_{d,n}$ as well as $\mathcal{T}^C_{d,n}$). We thus may safely assume that there are $S \geq (\log n)^{3/2}$ many root-to-pre-leaf x_1-free paths in T.

Now fix any set of S pre-leaf nodes in a complete binary tree of depth d. Fix a labeling of variables for each node on each of these paths except for the pre-leaves themselves (where the labeling is non-redundant and never uses x_1); call this labeling L. Fix any bit string $v = v_2, \ldots, v_n \in \{0,1\}^{n-1}$ corresponding to the parities of $N_{2,2}, \ldots, N_{n,2}$. We will show that conditioned on L being the labeling of the set of non-pre-leaf nodes on the x_1-free paths in T, the probability (over the random labeling of the S pre-leaf nodes) that each variable $x_2, \ldots, x_n$ occurs with the "right" parity $v_2, \ldots, v_n$ among the S pre-leaf nodes is $1/n^{\omega(1)}$. (In other words, we will show that regardless of what parities $N_{2,2}, \ldots, N_{n,2}$ have, the probability that each $N_{i,1}$ exactly matches the corresponding parity of $N_{i,2}$ is $1/n^{\omega(1)}$.) This suffices to establish (1). Thus, it suffices to prove the following lemma (the proof is given in Appendix D):

Lemma 19 *Fix $S \geq 2(\log n)^{3/2}$. For $i = 1, \ldots, S$ fix P_i to be some set of exactly $n-d$ "permissible" elements of $\{1, \ldots, n\}$. Fix $v = v_1, \ldots, v_n \in \{0,1\}^n$. Let P denote $P_1 \times P_2 \times \cdots \times P_S$, so $|P| = (n-d)^S$. Given $z = (z_1, \ldots, z_S) \in P$, for $j = 1, \ldots, n$ let $par_j(z)$ denote the number of occurrences modulo 2 of j in z. Then we have $\Pr_{z \in P}[par_j(z) = v_j \text{ for all } j = 1, \ldots, n] = \frac{1}{n^{\omega(1)}}$.*

We thus have the following balanced model analogues of Theorems 17 and 18.

Theorem 20 *Let $T \in \mathcal{T}^B_{d,n}$ where $d = \Theta(\log n), d \geq \frac{1}{2}\log n$. If x_1 is not a bottleneck then the probability that x_1 passes the test in line 8 is $1/n^{\omega(1)}$.*

Theorem 21 *For any n, any polynomial $p(\cdot)$, any $\delta > 1/p(n)$, and any $\epsilon > 0$, algorithm* **LearnTree** *will with probability at least $1 - \delta$ (over a random choice of tree T from $\mathcal{T}^B_{d,n}$ and the randomness of the example oracle) produce a hypothesis decision tree T' that ϵ-approximates the target with respect to the uniform distribution.* **LearnTree** *runs in time polynomial in n and $1/\epsilon$.*

9 Identifying bottlenecks in the uniform model $\mathcal{T}^U_{d,n}$

As in Section 7, it suffices to prove that if x_1 is not a bottleneck in a random tree T drawn from $\mathcal{T}^U_{d,n}$, then $\Pr[x_1 \text{ passes the test in line 8}] = 1/n^{\omega(1)}$. Our basic approach is similar to Section 7 but there are some differences as shown below.

We now let S be a random variable which denotes the number of x_1-free paths from the root to a pre-leaf of depth d in a random T drawn from $\mathcal{T}^U_{d,n}$. Note that in $\mathcal{T}^U_{d,n}$ it *is* possible to have $S = 0$ even if x_1 is not a bottleneck; this can happen if there exist x_1-free root-to-pre-leaf paths in T but all such pre-leaves have depth less than d. We will establish the following lemma in Appendix E:

Lemma 22 *For any value $0 \leq k \leq (\log n)^{3/2}$ we have $\Pr_{T \in \mathcal{T}^U_{d,n}}[S = k \mid x_1 \text{ is not a bottleneck }] = 1/n^{\omega(1)}$.*

With this lemma we can prove a uniform model analogue of Theorem 17:

Theorem 23 *Let $T \in \mathcal{T}^U_{d,n}$ where $d = \Theta(\log n), d \geq \frac{1}{2}\log n$. If x_1 is not a bottleneck then the probability that x_1 passes the test in line 8 is $1/n^{\omega(1)}$.*

Proof: Since x_1 is not a bottleneck, we have $S \geq (\log n)^{3/2}$ with probability $1 - \frac{1}{n^{\omega(1)}}$ by Lemma 22. We now observe that Lemma 16 holds identically (with the same proof) if in its statement T is drawn from $\mathcal{T}^U_{d,n}$ and we further condition on x_1 not being a bottleneck in T. We thus have that with $1 - \frac{1}{n^{\omega(1)}}$ probability there is a set C of $(\log n)^{5/4}$ variables with the properties stated in Lemma 16. The rest of the proof exactly follows the proof of Theorem 17. ■

Now a nearly[3] identical proof to that of Theorem 18 gives us our main learning result for uniform random trees:

Theorem 24 *For any n, any polynomial $p(\cdot)$, any $\delta > 1/p(n)$, and any $\epsilon > 0$, algorithm* **LearnTree** *will with probability at least $1 - \delta$ (over a random choice of tree T from $\mathcal{T}^U_{d,n}$ and the randomness of the example oracle) produce a hypothesis decision tree T' that ϵ-approximates the target with respect to the uniform distribution.* **LearnTree** *runs in time polynomial in n and $1/\epsilon$.*

10 Conclusions and Future Work

We have given positive results for learning several natural models of random log-depth decision trees under uniform. Many interesting questions remain about related models of average case learning:

- Can similar results be established for natural models of random decision trees of polynomial size (as opposed to logarithmic depth)?
- Can similar results be established for random DNF formulas or random monotone DNF?

[3]We need to fold the distribution irregularity noted in Section 5 into δ at the obvious point in the proof.

- Can our results be extended to learning under a broader class of distributions?
- Can similar ideas be used to learn with high probability when the target is drawn randomly from an interesting non-uniform distribution over log-depth trees?

It seems possible that progress in these directions could eventually lead to useful practical algorithms.

References

[1] A. Blum. Rank-r decision trees are a subclass of r-decision lists. *Information Processing Letters*, 42(4):183–185, 1992.

[2] A. Blum, M. Furst, M. Kearns, and R. Lipton. Cryptographic Primitives Based on Hard Learning Problems. In *Advances in Cryptology – CRYPTO '93*, pages 278–291, 1993.

[3] 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 Annual ACM Symposium on Theory of Computing*, pages 253–262, 1994.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification of Regression Trees.* Wadsworth, 1984.

[5] N. Bshouty. Exact learning boolean functions via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.

[6] A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

[7] T. Hancock. Learning $k\mu$ decision trees on the uniform distribution. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 352–360, 1993.

[8] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. *SIAM J. on Computing*, 22(6):1331–1348, 1993.

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

A Proof of Lemma 7

The proof is by induction on the depth d of T. If $d < 1$ then the only possible bottleneck in T is the root and the lemma holds vacuously. For the inductive step, consider a tree T of depth $d \geq 1$.

- If x_i is the root of T then T' is just T.
- If x_i is not the root of T then T has some variable x_j at the root with left subtree denoted by T_1 and right subtree T_2. Since x_i is a bottleneck in T but is not the root of T, x_i must be a bottleneck in T_1 and in T_2. By the induction hypothesis there are trees T_1', T_2' each of which has x_i at its root and such that each is structurally equivalent to T_1, T_2, respectively. By definition of structural equivalence, the tree T'' with x_j at its root and T_1', T_2' as the left and right child subtrees of the root is structurally equivalent to T. Performing a root swap on T'' gives the required T'.

■

B Proof of Lemma 12

We prove the lemma for the complete tree model $\mathcal{T}^C_{d,n}$; the proof holds unchanged for $\mathcal{T}^B_{d,n}$ and is easily adapted to $\mathcal{T}^U_{d,n}$. We have

$$\Pr[T \text{ is not read-}k] \quad \leq \quad n \cdot \Pr[x_1 \text{ occurs } k' > k \text{ times in } T \text{ for some } k'].$$

Since T is of depth r, there are $2^{r+1}-1$ occurrences of variables in T. The probability that x_1 occurs k' times in T is at most $\binom{2^{r+1}-1}{k'} \cdot \left(\frac{1}{n-r}\right)^{k'}$. This is because there are at most $\binom{2^{r+1}-1}{k'}$ possible sets of locations for the k' occurrences of x_1 in T, and for each location the probability that x_1 actually occurs there is at most $1/(n-r)$ (even conditioned on any labeling of the other nodes of the tree), since each location has at most r ancestors.

Since $\binom{a}{b} \leq a^b$ and $1/(n-r) < 2/n$, we can upper bound this probability by

$$2^{(r+1)k'} \cdot 2^{k'}/n^{k'} = 2^{(r+2)k'}/n^{k'} = n^{(1-\epsilon)k'}/n^{k'} = 1/n^{\epsilon k'} \leq 1/n^{\epsilon k}.$$

Since there are at most n possible values of $k' \geq k$ (no variable can occur more than n times since the size of the tree is less than n) all in all we have

$$\Pr_{T \in \mathcal{T}_{r,n}}[T \text{ is not read-}k] \leq 1/n^{\epsilon k - 2} = 1/n^C. \qquad \blacksquare$$

C Proofs of Lemmas 14, 15, and 16

We first prove Lemma 14. Let us write $p_{d,n}$ to denote the probability that $S = 0$ for a random T drawn from $\mathcal{T}^C_{d,n}$, i.e.

$$p_{d,n} = \Pr_{T \in \mathcal{T}^C_{d,n}}[x_1 \text{ is a bottleneck in } T].$$

Lemma 14 follows directly from the following:

Proposition 25 *For $0 \leq d \leq n-1$ we have $p_{d,n} \leq \frac{1}{n-d}$.*

Proof: Clearly $p_{0,n} = \frac{1}{n}$. For $d \geq 1, n \geq 1$ we have $p_{d,n} = \frac{1}{n} + \frac{n-1}{n}(p_{d-1,n-1})^2$. This is because with probability $1/n$ the root is x_1. With probability $\frac{n-1}{n}$ the root is some $x_j \neq x_1$ in which case each of the subtrees of the root is drawn from $\mathcal{T}^C_{d-1,n-1}$, and x_1 is a bottleneck iff it is a bottleneck in each of these two subtrees.

Fix any $m > 0$. We prove that for all $d \geq 0$ we have $p_{d,d+m} \leq \frac{1}{m}$; the proof is by induction on d. The base case holds since $p_{0,m} = \frac{1}{m}$. For the induction step, we have

$$\begin{aligned} p_{d+1,d+m+1} = \frac{1}{d+m+1} + \frac{d+m}{d+m+1}(p_{d,d+m})^2 \quad &\leq \quad \frac{1}{d+m+1} + \frac{d+m}{d+m+1}\left(\frac{1}{m}\right)^2 \\ &= \quad \frac{m^2+m+d}{m^2(m+d+1)}. \end{aligned}$$

This is at most $1/m$ iff $m^3+dm+m^2 \leq m^3+dm^2+m^2$ which is true since $m \geq 1$, so the proposition is proved. $\blacksquare$

We will prove Lemma 15 in a moment using Lemma 26 below. First, a definition and some introductory analysis.

Let $p_{k,d,n}$ denote $\Pr_{T \in \mathcal{T}^C_{d,n}}[S = k]$. For $k \geq 1$ and $d \geq 1$ we have

$$p_{k,d,n} = \frac{n-1}{n} \sum_{i=0}^{k} p_{i,d-1,n-1} p_{k-i,d-1,n-1}. \tag{2}$$

To see this, note that there are exactly k x_1-free root-to-preleaf paths in T iff (1) the root is some variable other than x_1, and (2) the left and right subtrees (each of which is drawn from $\mathcal{T}^C_{d-1,n-1}$) have i and $k-i$ x_1-free root-to-preleaf paths, respectively, for some $1 \leq i \leq k$. For the base cases, we have $p_{c,0,n} = 0$ for $c \geq 2$ since it is impossible to have two paths to pre-leaves in a tree of depth 0. $p_{1,0,n} = \frac{n-1}{n}$ since there is exactly one x_1-free path as long as the root is not x_1. $p_{0,0,n} = \frac{1}{n}$ by the same reasoning. Finally, $p_{0,d,n} \leq \frac{1}{n-d}$ by Lemma 14.

The following lemma is proved in section C.1:

Lemma 26 *Let $c = \Theta(\log n)$, $c \geq \frac{3}{8} \log n$ and $\ell \leq poly(n)$. Then*

$$p_{\ell,c,n} \leq t(n) + \sum_{j=1}^{(\log n)^{1/3}} \sum_{i=(1/4)\log n+1}^{\ell-(1/4)\log n-1} p_{i,c-j,n-j} \tag{3}$$

where $t(n) = 1/n^{\omega(1)}$.

Proof of Lemma 15. Recall that $k \leq (\log n)^{3/2}$, $d = \Theta(\log n)$ and $d \geq \frac{1}{2}\log n$. By Lemma 26 we have

$$p_{k,d,n} \leq t(n) + \sum_{j=1}^{(\log n)^{1/3}} \sum_{i=(1/4)\log n+1}^{k-(1/4)\log n-1} p_{i,d-j,n-j} \tag{4}$$

We now repeatedly apply Lemma 26 to the right-hand side of inequality (4). The key observation is that each time we apply Lemma 26 to bound some $p_{\ell,c,n'}$ by the right side of (3), the first subscript (ℓ) decreases by at least $\frac{1}{4}\log n$ in every new occurrence of $p_{\cdot,\cdot,\cdot}$. Hence the "depth" of this repeated replacement will be at most $4(\log n)^{1/2}$ (since $k \leq (\log n)^{3/2}$), at which point the summation over i in the right hand side of (3) will be empty.

We now observe that each application of Lemma 26 replaces one $p_{\cdot,\cdot,\cdot}$ with at most $(\log n)^{1/3} \cdot (\log n)^{3/2} < (\log n)^2$ new $p_{\cdot,\cdot,\cdot}$'s. Since the replacement depth is at most $4(\log n)^{1/2}$ and $t(n) = 1/n^{\omega(1)}$, it follows that

$$p_{k,d,n} \leq \frac{1}{n^{\omega(1)}} \cdot \left((\log n)^2\right)^{4(\log n)^{1/2}} = \frac{1}{n^{\omega(1)}} \cdot 2^{8(\log n)^{1/2}\log\log n} = \frac{1}{n^{\omega(1)}}$$

and Lemma 15 is proved. ■

Now we prove Lemma 16.

Proof of Lemma 16. Fix any set R of $(\log n)^{3/2}$ pre-leaf nodes in a complete binary tree structure of depth d. Fix any non-redundant labeling of all of the ancestors of all of these pre-leaves which does not use x_1 anywhere. Now each labeling of the nodes in R which does not use x_1 and maintains non-redundancy is equally likely under the conditioning of the lemma. Note that for each node in R there are $n-d-1$ legal labelings (since the label must not use x_1 or any of the d ancestors of the node).

Consider a random legal labeling of the nodes in R. Partition the nodes of R into $(\log n)^{5/4}$ disjoint subsets $R_1, \ldots, R_{(\log n)^{5/4}}$ each of size $(\log n)^{1/4}$. Let F denote a set of "forbidden" labels;

initially F is the set of all variables which label ancestors of nodes in R (plus x_1). Let F_0 denote the size of this initial set, so initially we have $|F| = F_0 \leq 1 + d(\log n)^{3/2} = O((\log n)^{5/2})$. We consider the subsets $R_1, \ldots$ in turn. The probability that every node in R_1 is assigned a forbidden label is at most $\left(\frac{F_0}{n-d-1}\right)^{(\log n)^{1/4}} = 1/n^{\omega(1)}$. Thus we may suppose that there is some pre-leaf $v_1 \in R_1$ which receives a non-forbidden label; we add this label to F. Now the probability that every node in R_2 receives a forbidden label is at most $\left(\frac{F_0+1}{n-d-1}\right)^{(\log n)^{1/4}} = 1/n^{\omega(1)}$, so we may suppose that there is some pre-leaf $v_2 \in R_2$ which receives a non-forbidden label; we add this label to F. Continuing in this fashion for $(\log n)^{5/4}$ steps, and noting that $|F|$ never exceeds $O((\log n)^{5/2})$, we have that with probability $1-1/n^{\omega(1)}$ there is a set $v_1, \ldots, v_{(\log n)^{5/4}}$ of nodes each of which receives a non-forbidden label. This set is easily seen to satisfy the desired conditions for C. ■

C.1 Proof of Lemma 26

Our proof of Lemma 26 will use the following intermediate lemma. Note that we allow a slightly weaker bound on d than usual in this lemma; we will need this slightly weaker bound later.

Lemma 27 *For any value $1 \leq k \leq \frac{1}{4}\log n$ and any value $d \geq \frac{1}{3}\log n$ we have $p_{k,d,n} = \Pr_{T \in \mathcal{T}^C_{d,n}}[S = k] = 1/n^{\omega(1)}$.*

Proof: We first consider the case $k = 1$. There are exactly 2^d possible locations (pre-leaves) where an x_1-free path from the root to a pre-leaf could end. Consider any such location. In order for this to be the only x_1-free path to a pre-leaf in T, it must be the case that every node on this path (except the root) has the property that the subtree rooted at its sibling has x_1 as a bottleneck. These d subtrees are clearly disjoint; the one at depth ℓ is drawn from $\mathcal{T}^C_{d-\ell,n-\ell}$ (over a suitable set of $n - \ell$ variables which includes x_1 since the path is x_1-free) and hence by Lemma 14 each subtree has x_1 as a bottleneck with probability at most $\frac{2}{n}$. Thus $\Pr[S = 1]$ is at most $2^d \cdot \left(\frac{2}{n}\right)^d = \left(\frac{4}{n}\right)^d$ which is $1/n^{\omega(1)}$ since $d \geq \frac{1}{3}\log n$.

The general case for any $1 \leq k \leq \frac{1}{4}\log n$ is similar. We use the following fact which we prove later:

Fact 28 *Fix any set of k root-to-preleaf paths in T. Let N be the number of subtrees of T which are rooted at an internal node and (1) are not rooted on any of these k paths, but (2) have their parent on one of these k paths. Then $N \geq d - \log k$.*

There are $\binom{2^d}{k}$ possible sets of k pre-leaves where the x_1-free paths might end. As in the case $k = 1$, each subtree as in Fact 28 must have x_1 as a bottleneck, but as in the $k = 1$ case each such subtree has x_1 as a bottleneck with probability at most $2/n$. Thus the probability that $S = k$ is at most (by Fact 28)

$$\binom{2^d}{k} \cdot \left(\frac{2}{n}\right)^{d-\log k} \leq 2^{dk}\left(\frac{2}{n}\right)^{d-\log k} \leq n^{d/4} \cdot \left(\frac{2}{n}\right)^d \cdot n^{\log k} = n^{\log k}\left(\frac{2}{n^{3/4}}\right)^d,$$

where the second inequality uses $k \leq \frac{1}{4}\log n$. This is $1/n^{\omega(1)}$ since $d \geq \frac{1}{3}\log n$ and $k \leq \frac{1}{4}\log n$. ■

Proof of Fact 28. It is clear that there are exactly $2^d - k$ pre-leaves contained in the desired subtrees of T. Each subtree contains 2^i of these pre-leaves for some i, and clearly different subtrees have disjoint sets of pre-leaves. Since the binary representation of $2^d - k$ starts with $d - \log k$ ones, there must be at least $d - \log k$ such subtrees (it is impossible to add up t powers of 2 and get a binary number with more than t ones). ■

Now we prove Lemma 26. From the recursive equation (2) we have

$$\begin{aligned} p_{\ell,c,n} &\leq 2p_{0,c-1,n-1}p_{\ell,c-1,n-1} + \sum_{i=1}^{\ell-1} p_{i,c-1,n-1}p_{\ell-i,c-1,n-1} \\ &\leq \frac{4}{n}p_{\ell,c-1,n-1} + \sum_{i=1}^{\ell-1} p_{i,c-1,n-1}p_{\ell-i,c-1,n-1} \end{aligned} \tag{5}$$

where the last inequality holds (with room to spare) by Lemma 14 since $c = \Theta(\log n) < n/2$. Repeatedly applying (5) we have

$$\begin{aligned} p_{\ell,c,n} &\leq \left(\frac{4}{n}\right)^2 p_{\ell,c-2,n-2} + \frac{4}{n}\sum_{i=1}^{\ell-1} p_{i,c-2,n-2}p_{\ell-i,c-2,n-2} + \sum_{i=1}^{\ell-1} p_{i,c-1,n-1}p_{\ell-i,c-1,n-1} \\ &\leq \cdots \\ &\leq \left(\frac{4}{n}\right)^c p_{\ell,0,n-c} + \sum_{j=1}^{c}\left(\frac{4}{n}\right)^{j-1}\sum_{i=1}^{\ell-1} p_{i,c-j,n-j}p_{\ell-i,c-j,n-j}. \end{aligned}$$

Since each value $p_{\cdot,\cdot,\cdot}$ is a probability it is easy to see that for any value of j the inner sum over i is at most ℓ =poly(n). Recalling that $c \geq \frac{3}{8}\log n$, we may truncate the sum over j at (say) $(\log n)^{1/3}$ and thus have

$$\begin{aligned} p_{\ell,c,n} &\leq \frac{1}{n^{\omega(1)}} + \sum_{j=1}^{(\log n)^{1/3}}\sum_{i=1}^{\ell-1} p_{i,c-j,n-j}p_{\ell-i,c-j,n-j} \\ &\leq \frac{1}{n^{\omega(1)}} + \sum_{j=1}^{(\log n)^{1/3}}\left[2\sum_{i=1}^{(1/4)\log n} p_{i,c-j,n-j} + \sum_{i=(1/4)\log n+1}^{\ell-(1/4)\log n-1} p_{i,c-j,n-j}\right]. \end{aligned}$$

Since $c - (\log n)^{1/3}$ is at least $(1/3)\log n$, Lemma 27 implies that the first sum over i inside the brackets is $1/n^{\omega(1)}$ for all $j = 1, \ldots, (\log n)^{1/3}$. We thus have

$$p_{\ell,c,n} \leq \frac{1}{n^{\omega(1)}} + \sum_{j=1}^{(\log n)^{1/3}}\sum_{i=(1/4)\log n+1}^{\ell-(1/4)\log n-1} p_{i,c-j,n-j}$$

as desired, and Lemma 26 is proved.

D Proof of Lemma 19

We say that $z \in P$ *yields* $v \in \{0,1\}^n$ if $par_j(z) = v_j$ for all j. Let $V \subseteq P$ denote the set of all z which yield v; thus our goal is to show that $|V|/|P| = 1/n^{\omega(1)}$. We do this by defining a mapping M with the following properties:

1. M assigns to each $z \in V$ a set of $(n - d - 2\log n)^{\log n}$ strings all of which are in $P - V$.

2. For each $z \in V$, for any $x \in M(z)$, the number of $z' \in V$ such that $x \in M(z')$ is at most $(1/n^{\omega(1)}) \cdot (n - d - 2\log n)^{\log n}$.

These properties imply that $|V|/|P - V| = 1/n^{\omega(1)}$ which establishes the lemma.

The mapping M is defined as follows: given $z \in V$, the elements of $M(z)$ are those $x \in P$ which satisfy the following conditions:

- $x_i = z_i$ for all $i = 1, \dots, S - \log n$.
- For $i = S - \log n + 1, \dots, S$, coordinate x_i is an element of P_i which (a) does not occur in the last $\log n$ coordinates of z, and (b) does not equal x_j for any other value $j \in S - \log n + 1, \dots, S$.

There are at least $(n - d - 2\log n)^{\log n}$ elements of $M(z)$ since the second condition above rules out at most $2\log n$ of the $n - d$ elements of P_i for each i.

Fix some $z \in V$ and some $x \in M(z)$. We now show that there are few $z' \in V$ such that $x \in M(z')$. Note first that in order for z' to have $x \in M(z')$ it must be the case that $z'_i = z_i$ for all $i = 1, \dots, S - \log n$. Let z^* denote this $S - \log n$ character prefix $z_1 \dots z_{S-\log n}$. We now show that only a small number of the $(n-d)^{\log n}$ possible completions $\sigma \in P_{S-\log n+1} \times \cdots \times P_S$ will be such that $z' = z^*\sigma$ belongs to V.

We prove this by showing that in fact only a small number of the $n^{\log n}$ possible completions $\sigma \in \{1, \dots, n\}^{\log n}$ will be such that $z^*\sigma$ belongs to V. To see this, note that in order for $z^*\sigma$ to belong to V, each $j \in \{1, \dots, n\}$ must occur either an even or odd number of times in σ (depending on whether $par_j(z^*)$ does or does not match v_j). Now we observe that the probability that these n parity conditions are all satisfied by a random $\sigma \in \{1, \dots, n\}^{\log n}$ is precisely the probability that a uniform random walk in the Boolean cube $\{0,1\}^n$ ends up, after precisely $\log n$ steps, at some particular vertex $w \in \{0,1\}^n$ (where the walk starts at 0^n and proceeds to a randomly chosen neighbor of the current node at each step). Since $d = \Theta(\log n)$ we have $(n - d - 2\log n)^{\log n}/n^{\log n} = \Theta(1)$, and thus Lemma 19 follows from the following elementary fact, which for completeness we now prove:

Proposition 29 *For all $w \in \{0,1\}^n$, the probability that a uniform random walk of precisely $\log n$ steps starting at 0^n ends at w is $1/n^{\omega(1)}$.*

Proof: Let $|w|$ denote the number of nonzero coordinates in w. Let p_w denote the probability that the walk ends at w. We first observe that if $|w| = |w'|$ then by symmetry $p_w = p_{w'}$. Thus for any w such that $|w| \geq \frac{1}{3}\log n$ we clearly have $p_w \leq 1/\binom{n}{(1/3)\log n} = 1/n^{\omega(1)}$. Thus we may suppose that $|w| \leq \frac{1}{3}\log n$. Any walk of $\log n$ steps which ends at such a w must select at most $\frac{2}{3}\log n$ distinct indices from $\{1, \dots, n\}$ in total, since at most $\frac{1}{3}\log n$ of the indices selected are selected exactly once. The number of possible $\log n$ step walks which select at most $\frac{2}{3}\log n$ distinct indices is at most $\binom{n}{(2/3)\log n} \cdot (2/3\log n)^{\log n}$, since there are $\binom{n}{(2/3)\log n}$ ways to choose the selected indices and then $(2/3\log n)^{\log n}$ ways to choose the walk using these indices. This is less than $n^{2/3\log n} \cdot (n^{1/6})^{\log n} = n^{(5/6)\log n}$ and hence such walks occur with total probability at most $1/n^{(1/6)\log n}$ since there are $n^{\log n}$ possible walks in total. ■

E Proofs of Lemmas 22, 9, and 11

Before proving Lemma 22 we first establish some useful notation and observations.

Let $C_{d,n}$ denote the number of non-redundant decision trees over $\{x_1, \dots, x_n\}$ of depth at most d. (So $C_{-1,n} = 2$ since a single leaf can be either -1 or 1; $C_{0,n} = 2 + 4n$ since there are $4n$ possibilities for a depth-0 tree depending on the variable at the root and the two leaf bits; etc.) The following observation will be useful:

Observation 30 *Drawing a random T from $\mathcal{T}_{d,n}^U$ is equivalent to generating T via the following randomized process which we call* $\texttt{MakeTree}_{d,n}$*:*

- *With probability $1/C_{d,n}$ take T to be the one-node tree $+1$ and halt. Likewise, with probability $1/C_{d,n}$ take T to be the one-node tree -1 and halt.*
- *With probability $1-2/C_{d,n}$ pick a random variable from $x_1,\ldots,x_n$ as the root of T. Construct its left and right subtrees by independently performing two calls to* $\texttt{MakeTree}_{d-1,n-1}$*, using for each call the set of $n-1$ variables which were not selected for the root.*

We write $q_{d,n}$ to denote $1-\frac{2}{C_{d,n}}$, i.e. $q_{d,n}$ is the probability that a randomly drawn T from $\mathcal{T}_{d,n}^U$ is nontrivial (recall that there are exactly two trivial trees).

We now write $p_{d,n}$ to denote the probability that a random T drawn from $\mathcal{T}_{d,n}^U$ has x_1 as a bottleneck:

$$p_{d,n} \equiv \Pr_{T\in\mathcal{T}_{d,n}^U}[x_1 \text{ is a bottleneck}].$$

We have the following analogue of Lemma 14:

Lemma 31 *For all $0 \le d \le n-1$, $p_{d,n} \le \frac{1}{n-d}$.*

Proof: The proof differs only slightly from that of Lemma 14. We now have $p_{0,n} = \frac{4}{4n+2} < \frac{1}{n}$ since exactly four of the $4n+2$ trees of depth at most 0 have x_1 as a bottleneck (i.e. as the root). For $d \ge 1, n \ge 1$ we now have

$$p_{d,n} = q_{d,n}\left(\frac{1}{n} + \frac{n-1}{n}(p_{d-1,n-1})^2\right) \tag{6}$$

To see this, note that x_1 is a bottleneck only if T is nontrivial, which occurs with probability $q_{d,n}$. If T is nontrivial then with probability $\frac{1}{n}$ the root is x_1 in which case x_1 is a bottleneck. Otherwise, x_1 is a bottleneck if and only if x_1 is a bottleneck in the left and right subtrees of T, each of which is drawn from $\mathcal{T}_{d-1,n-1}^U$.

Comparing the initial conditions and recurrence relation for $p_{d,n}$ with those of Lemma 14 it is clear that the current $p_{d,n}$ is dominated by the earlier recurrence, and the lemma is proved. ■

Now we can prove Lemma 22. We have that

$$\begin{aligned}
\Pr_{T\in\mathcal{T}_{d,n}^U}[S=k \mid x_1 \text{ is not a bottleneck}] &= \frac{\Pr[S=k \;\&\; x_1 \text{ is not a bottleneck}]}{\Pr[x_1 \text{ is not a bottleneck}]} \\
&< 2\Pr[S=k \;\&\; x_1 \text{ is not a bottleneck}] \\
&\le 2\Pr[S=k]
\end{aligned}$$

where the first inequality is by Lemma 31. Thus it suffices to prove the following two lemmas:

Lemma 32 *For all $1 \le k \le (\log n)^{3/2}$ we have $\Pr_{T\in\mathcal{T}_{d,n}^U}[S=k] = 1/n^{\omega(1)}$.*

Lemma 33 *For $d \ge \frac{1}{2}\log n$, $d = \Theta(\log n)$, $\Pr_{T\in\mathcal{T}_{d,n}^U}[S=0 \mid x_1$ is not a bottleneck$] = 1/n^{\omega(1)}$.*

(Note that bounding $\Pr[S=0 \mid x_1 \text{ is not a bottleneck}]$ by $2\Pr[S=0]$ is a bad idea since $S=0$ whenever x_1 is a bottleneck and this occurs with probability roughly $1/n$; hence we use the above approach of handling $S=0$ separately by bounding the conditional probability directly.)

Proof of Lemma 32. We closely imitate the proof of Lemma 15. Let $p_{k,d,n}$ now denote $\Pr_{T \in \mathcal{T}^U_{d,n}}[S = k]$. For $k \geq 1$ and $d \geq 1$ we now have

$$p_{k,d,n} = q_{d,n} \cdot \frac{n-1}{n} \cdot \sum_{i=0}^{k} p_{i,d-1,n-1} p_{k-i,d-1,n-1} \tag{7}$$

(The $q_{d,n} \cdot \frac{n-1}{n}$ is present because in order for S to be nonzero it must be the case that T is nontrivial and that the root is not x_1. If this is the case then the left and right subtrees (which, under this conditioning, are drawn from $\mathcal{T}^U_{d-1,n-1}$) must have i and $k-i$ x_1-free paths from their respective roots to pre-leaves at depth $d-1$ in those subtrees.) As before we have $p_{c,0,n} = 0$ for $c \geq 2$ since there cannot be two paths to pre-leaves in a depth-0 tree. We have $p_{1,0,n} = q_{0,n} \cdot \frac{n-1}{n}$ since there is one x_1-free path if and only if the tree is nontrivial and the root is not x_1. Moreover, we have $p_{0,0,n} = \frac{6}{4n+2}$ since the only trees of depth at most 0 which have $S = 0$ are the four trees with x_1 at the root and the two trivial trees.

Finally, if $k = 0$ and $d > 0$ then we have

$$p_{0,d,n} = (1 - q_{d,n}) + q_{d,n} \cdot \frac{1}{n} + q_{d,n} \cdot \frac{n-1}{n} (p_{0,d-1,n-1})^2. \tag{8}$$

(We have that $S = 0$ if T is trivial, or if T is nontrivial and x_1 is the root, or if T is nontrivial, some other variable is the root, and both subtrees have no x_1-free paths to pre-leaves at depth $d-1$.) Equation (8), combined with the fact that $1 - q_{d,n} < \frac{1}{n}$ for $d \geq 0$, implies that for $d \geq 1$ we have

$$p_{0,d,n} \leq \frac{2}{n} + \frac{n-1}{n} (p_{0,d-1,n-1})^2.$$

A straightforward analysis of this recurrence shows that $p_{0,d,n} < \frac{3}{n}$ for all $d = O(\log n)$.

Now let $\tilde{p}_{k,d,n}$ be defined by the same base case conditions (for $k = 0$ or $d = 0$) that we have just given, but be defined for $k \geq 1, d \geq 1$ by the rule from Section C, i.e.

$$\tilde{p}_{k,d,n} = \frac{n-1}{n} \sum_{i=0}^{k} \tilde{p}_{i,d-1,n-1} \tilde{p}_{k-i,d-1,n-1}.$$

(Note that these base conditions differ only slightly from the base conditions on $p_{k,d,n}$ in Section C; the bound $\frac{3}{n}$ which we have on $\tilde{p}_{0,d,n}$ is slightly weaker than the $\frac{2}{n}$ bound we used in the earlier proof, and the $\tilde{p}_{0,0,n}$ bound of $\frac{6}{4n+2}$ is slightly weaker than the old bound of $\frac{1}{n}$.) A proof entirely similar to that of Lemma 15 now establishes that $\tilde{p}_{k,d,n} = 1/n^{\omega(1)}$ for $1 \leq k \leq (\log n)^{3/2}$ and $d \geq \frac{1}{2} \log n$, $d = \Theta(\log n)$. We now observe that the recurrence for $\tilde{p}_{k,d,n}$ dominates the recurrence which we have in this section for $p_{k,d,n}$, and hence $p_{k,d,n} \leq \tilde{p}_{k,d,n} = 1/n^{\omega(1)}$ as well. (Note that we cannot argue directly that the old $p_{k,d,n}$ recurrence dominates the new one, since some initial values of the new recurrence are as remarked earlier slightly higher than the old values.) This proves Lemma 32. ■

It remains to prove Lemma 33.

Proof of Lemma 33. We have that

$$\begin{aligned} \Pr_{T \in \mathcal{T}^U_{d,n}}[S = 0 \mid x_1 \text{ is not a bottleneck}] &= \frac{\Pr[S = 0 \ \& \ x_1 \text{ is not a bottleneck}]}{\Pr[x_1 \text{ is not a bottleneck}]} \\ &< 2 \Pr[S = 0 \ \& \ x_1 \text{ is not a bottleneck}] \end{aligned}$$

since $\Pr[x_1 \text{ is not a bottleneck}] = 1 - p_{d,n} \geq \frac{1}{2}$ by Lemma 31. Since $S = 0$ whenever x_1 is a bottleneck, we have that

$$\Pr[S = 0 \;\&\; x_1 \text{ is not a bottleneck}] = \Pr[S = 0] - \Pr[x_1 \text{ is a bottleneck}].$$

Thus it suffices to bound $\epsilon_{d,n} \equiv p_{0,d,n} - p_{d,n}$. Combining Equations (6) and (8) we have

$$\begin{aligned}
\epsilon_{d,n} &= (1 - q_{d,n}) + q_{d,n} \cdot \frac{n-1}{n}\left[(p_{0,d-1,n-1})^2 - (p_{d-1,n-1})^2\right] \\
&= (1 - q_{d,n}) + q_{d,n} \cdot \frac{n-1}{n}\left[(p_{0,d-1,n-1} - p_{d-1,n-1})(p_{0,d-1,n-1} + p_{d-1,n-1})\right] \\
&= (1 - q_{d,n}) + q_{d,n} \cdot \frac{n-1}{n}\left[\epsilon_{d-1,n-1}(p_{0,d-1,n-1} + p_{d-1,n-1})\right].
\end{aligned}$$

Our bounds on $p_{0,d,n}$ and $p_{d,n}$ imply that $p_{0,d-1,n-1} + p_{d-1,n-1} \leq \frac{5}{n-1}$. The above thus implies

$$\epsilon_{d,n} \leq (1 - q_{d,n}) + \epsilon_{d-1,n-1} \cdot \frac{5}{n-1}.$$

It is clear that for $\ell \geq \frac{1}{4}\log n$, we have $1 - q_{\ell,n} = \frac{2}{C_{\ell,n}} = 1/2^{n^{\Omega(1)}}$. Since $\epsilon_{\frac{1}{4}\log n, n-d+\frac{1}{4}\log n}$ is clearly at most 1, expanding out the above recurrence we thus have

$$\epsilon_{d,n} < \frac{d - \frac{1}{4}\log n}{2^{n^{\Omega(1)}}} + \left(\frac{10}{n}\right)^{d - \frac{1}{4}\log n}.$$

Since $d \geq \frac{1}{2}\log n$, $d = \Theta(\log n)$ this is $\frac{1}{n^{\omega(1)}}$, and the lemma is proved. ■

Finally, we can also now prove two lemmas used in Section 5.

Proof of Lemma 11. It is easy to see that $C_{t,\Omega(n)} = 2^{n^{\Omega(1)}}$ for $t \geq \frac{1}{4}\log n$. Also, there are only $\mathrm{poly}(n)$ chances for `MakeTree` to output a trivial tree down to depth $d - \frac{1}{4}\log n$. Therefore, the chance of any leaf being output before depth $\frac{1}{4}\log n$ is at most $2^{n^{\Omega(1)}}$. ■

Proof of Lemma 9. In all three models we have that if $d = O(\log n)$ then $p_{d,n} < 2/n$. If T is drawn according to $\mathcal{T}_{d,n}$ and has a bottleneck variable x_1 labeling node v at depth $k \geq \frac{1}{8}\log n$, then the sibling of v and the sibling of each of v's non-root ancestors must have x_1 as a bottleneck. These probabilities are all independent, and each is at most $(2/n)$. ■