Learning random log-depth decision trees under the uniform distribution

Jeffrey C. Jackson*[1] and Rocco A. Servedio[2]

[1] Department of Mathematics and Computer Science
Duquesne University, Pittsburgh, PA 15282
jackson@mathcs.duq.edu
[2] Department of Computer Science
Columbia University, New York, NY 10027, USA
rocco@cs.columbia.edu

Abstract. We consider three natural models of random log-depth decision trees. We give an efficient algorithm that for each of these models learns—as a decision tree—all but an inverse polynomial fraction of such trees using only uniformly distributed random examples.

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., [3, 8]). 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 [2].

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 [5] 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. Kushilevitz and Mansour [7] 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 [4] 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

* This material is based upon work supported by the National Science Foundation under Grant No. CCR-0209064.

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.

In this work we propose a third approach to coping with the difficulty of learning decision trees: looking at the average case. Specifically, since we have been unable to design algorithms which can learn *all* decision trees, we focus instead on algorithms which can efficiently learn "average" decision trees. Another difference between our approach and some of the earlier theoretical work is that the hypothesis produced by our algorithm is a decision tree. A limitation of our algorithm is that we assume that examples are uniformly distributed.

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 7 contain the proofs of correctness for the learning algorithm. We conclude in Section 8.

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.

Due to the space restrictions in these proceedings, here we prove our main learning result (Theorem 2) only for the complete model $\mathcal{T}^C_{d,n}$. We have proved exact analogues of this result for for the uniform and balanced models $\mathcal{T}^U_{d,n}$ and $\mathcal{T}^B_{d,n}$ as well; the proofs for these models will be contained in the full version of the paper.

We assume throughout that d is $O(\log n)$, and that the learning algorithm knows the exact value of d. This latter assumption is w.l.o.g. 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 [7] 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 [7] directly implies:

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

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

(Note that this corollary implies that any tree of depth d has each $\hat{f}(e_i)$ of the form $i/2^d$ for some integer i. 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.)

From this we easily obtain:

Corollary 2. *For every decision tree f of depth d and every $1 \leq i \leq n$, there is an integer k such that $\hat{f}(e_i) = \frac{2k+1}{2^d}$ if and only if the total number of nodes/pairs of leaves satisfying the following conditions is odd for x_i:*

[1] [7] considered decision trees in which internal nodes can contain arbitrary parity functions; however as noted earlier we only allow single variables at internal nodes.

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.*

We say that any first-order Fourier coefficient of the form $\frac{2k+1}{2^d}$ is an *odd coefficient*, and all other first-order coefficients are *even coefficients*. Conditions 2 and 3 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 easily from the above corollaries, is:

Lemma 1. *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 $\Pr[\forall x_i \in S\ \hat{f}(e_i)$ 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 Corollary 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 A of m labeled pairs $\langle a, f(a)\rangle$ is—with probability at least $1-\delta$ over the choice of A—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 A, $\hat{f}(e_i) = R\left((\sum_{a\in A} a_i f(a))/m\right)$ where $R(\cdot)$ represents "rounding" the argument to the nearest rational number having denominator 2^d. Therefore, we also have

Corollary 3 (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 (Figure 1) 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

LearnTree$(n, d, \delta, \epsilon, EX(T,\mathcal{U}))$

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 $b \in \{-1, 1\}$.
10. **endif**
11. **endfor**
12. **return** "fail"

Fig. 1. Algorithm for learning random decision trees.

phase (lines 1-3) we employ an algorithm **UnikDTLearn**$(n, \epsilon, \delta, EX(T,\mathcal{U}))$ due to Hancock [6] to learn these remaining "shallow" decision trees.

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 Corollary 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 1.) Furthermore, we will argue in section 6 that Hancock's **UnikDTLearn** efficiently learns random trees of depth at most $\frac{1}{2}\log n$, 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 the complete random tree model $\mathcal{T}^C_{d,n}$.

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 obvious difficulty is that there may be a tree T' that computes the same function as T and that has the same depth but that has a different root variable; consider any tree representing $x_1 \oplus x_2$, for example. We will therefore be content with finding any of a set of "root-like" variables of T, which we call *bottlenecks*:

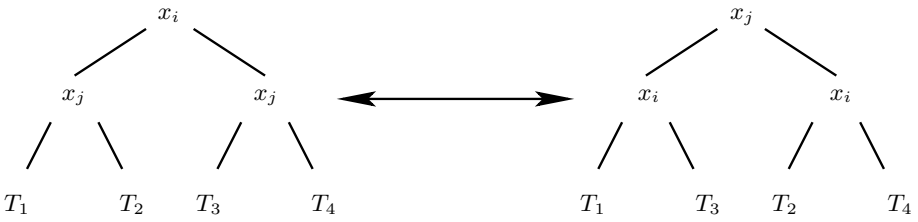

Fig. 2. Replacing the left structure by the right structure, or vice versa, is a swap.

Definition 1. *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 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 if a bottleneck variable is chosen as the root at each stage of the recursion and **FCExact** returns accurate values for all first-order Fourier coefficients then at every stage any bottleneck variable will pass the test at line 8 of **LearnTree**.

In later sections we will show that for a random tree T drawn from $\mathcal{T}^C_{d,n}$, 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 $\mathcal{T}^C_{d-1,n-1}$ (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 $\mathcal{T}^C_{d,n}$) 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 $\mathcal{T}^C_{d-1,n-1}$.

We now show that as long as any bottleneck is chosen, the two resulting subfunctions do indeed correspond to draws from $\mathcal{T}^C_{d-1,n-1}$. While the two draws are not independent of each other, this does not negatively impact the algorithm.

We say that two decision trees T and T' are *structurally equivalent* if T' can be obtained from T by performing a sequence of "swaps" of subtrees, where a swap is a replacement of one subtree by another subtree as shown in Figure 2. Note that if two trees are structurally equivalent then they compute the same function and have the same depth. The following lemma, which can be proved inductively (omitted due to space limitations), shows that any bottleneck variable can be swapped to the root of a tree in a way that preserves structural equivalence.

Lemma 2. *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 (implicit in the proof Lemma 2) 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 3. *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 $1 \leq i, d \leq n$, $\mathcal{T}^{-1}_{d-1,n-1}$ and $\mathcal{T}^1_{d-1,n-1}$ are both identical to $\mathcal{T}^C_{d-1,n-1}$.*

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. □

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 $\mathcal{T}^C_{d,n}$-random trees of depth $\frac{1}{2}\log n$ with high probability. We address the second point first in the next section.

[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.

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 [6].

Recall that a decision tree T is *read-k* if each variable labels at most k nodes in T. The following lemma is easily proved (proof omitted due to space):

Lemma 4. *Let $r = ((1-\epsilon)\log n) - 2$ for some constant $\epsilon > 0$. Let C be any constant. Then we have $\Pr_{T \in \mathcal{T}^C_{r,n}}[T$ 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 $\mathcal{T}^C_{d,n}$ 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 [6] 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 5. *If the oracle $EX(T,\mathcal{U})$ in the call to* **UnikDTLearn** *in* **LearnTree** *represents a tree T distributed according to $\mathcal{T}^C_{d,n}$, then* **UnikDTLearn** *returns a decision tree that ϵ-approximates T with probability (over the random choice of T and the randomness in $EX(T,\mathcal{U})$) 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 $d = \Theta(\log n), d \geq \frac{1}{2}\log n$.

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 1 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 $S > 0$ iff x_1 is not a bottleneck). We will prove the following lemmas in Appendix A.

Lemma 6. *For $0 \leq d \leq n-1$, $\Pr_{T \in \mathcal{T}^C_{d,n}}[S = 0] \leq \frac{1}{n-d}$.*

Lemma 7. *For any value $1 \leq k \leq (\log n)^{3/2}$ we have $\Pr[S = k] = 1/n^{\omega(1)}$.*

Lemma 8. *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 1. *Let $T \in \mathcal{T}^C_{d,n}$ 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 $S = 0$ iff x_1 is a bottleneck, we have

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

where the last equality follows from Lemmas 6 and 7. Thus we may assume that $S \geq (\log n)^{3/2}$. Lemma 8 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 1, 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 2. *For any $d = O(\log n)$, 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 $\mathcal{T}^C_{d,n}$ 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 $1/\epsilon$.*

Proof. By Lemma 5, 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 $\mathcal{T}^C_{c,n}$ 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, if the initial tree is distributed according to $\mathcal{T}^C_{d-k,n-k}$ then the two functions obtained by restricting on either value of this bottleneck variable will both be distributed according to $\mathcal{T}^C_{d-k-1,n-k-1}$. 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 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 monotone DNF?
- Can our results be extended to learning under broader classes of distributions, either over examples or over 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, 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.
[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification of Regression Trees*. Wadsworth, 1984.
[4] N. Bshouty. Exact learning boolean functions via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.
[5] A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.
[6] 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.
[7] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. on Computing*, 22(6):1331–1348, 1993.
[8] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

A Proofs of Lemmas 6, 7, and 8

We first prove Lemma 6. 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 6 follows directly from the following:

Proposition 1. *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

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

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. □

We will prove Lemma 7 in a moment using using Lemma 9 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

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

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 $0 \leq i \leq k$. 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 6.

The following lemma is proved in section A.1:

Lemma 9. *Let $c = \Theta(\log n), c \geq \frac{3}{8}\log n$ and $\ell \leq$ poly(n). Then*

$$p_{\ell,c,n} \leq t(n) + \sum_{j=1}^{(\log n)^{1/3}} \sum_{i=(1/4)\log n+1}^{\ell-(1/4)\log n-1} p_{i,c-j,n-j} \tag{2}$$

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

Proof of Lemma 7. Recall that $k \leq (\log n)^{3/2}$, $d = \Theta(\log n)$ and $d \geq \frac{1}{2}\log n$. By Lemma 9 we have

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

We now repeatedly apply Lemma 9 to the right-hand side of inequality (3). The key observation is that each time we apply Lemma 9 to bound some $p_{\ell,c,n'}$ by the right side of (2), 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 (2) will be empty.

We now observe that each application of Lemma 9 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 7 is proved. □

Now we prove Lemma 8.

Proof of Lemma 8. 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. □

A.1 Proof of Lemma 9

Our proof of Lemma 9 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 10. *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 6 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 3 *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 3 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 3)

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

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

Proof of Fact 3. 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 9. From the recursive equation (1) we have

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

where the last inequality holds (with room to spare) by Lemma 6 since $c = \Theta(\log n) < n/2$. Repeatedly applying (4) we have

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

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 $\ell = \text{poly}(n)$. Recalling that $c \ge \frac{3}{8}\log n$, we may truncate the sum over j at (say) $(\log n)^{1/3}$ and thus have

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

Since $c - (\log n)^{1/3}$ is at least $(1/3)\log n$, Lemma 10 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

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

as desired, and Lemma 9 is proved.