Lifting uniform learners via distributional decomposition

Guy Blanc *Stanford*
Jane Lange *MIT*
Ali Malik *Stanford*
Li-Yang Tan *Stanford*

March 31, 2023

Abstract

We show how any PAC learning algorithm that works under the uniform distribution can be transformed, in a blackbox fashion, into one that works under an arbitrary and unknown distribution $\mathcal{D}$. The efficiency of our transformation scales with the inherent complexity of $\mathcal{D}$, running in $\mathrm{poly}(n, (md)^d)$ time for distributions over $\{\pm 1\}^n$ whose pmfs are computed by depth-d decision trees, where m is the sample complexity of the original algorithm. For monotone distributions our transformation uses only samples from $\mathcal{D}$, and for general ones it uses subcube conditioning samples.

A key technical ingredient is an algorithm which, given the aforementioned access to $\mathcal{D}$, produces an *optimal* decision tree decomposition of $\mathcal{D}$: an approximation of $\mathcal{D}$ as a mixture of uniform distributions over disjoint subcubes. With this decomposition in hand, we run the uniform-distribution learner on each subcube and combine the hypotheses using the decision tree. This algorithmic decomposition lemma also yields new algorithms for learning decision tree distributions with runtimes that exponentially improve on the prior state of the art—results of independent interest in distribution learning.

1 Introduction

A major strand of research in learning theory concerns the learning of an unknown target function $f : \{\pm 1\}^n \to \{0, 1\}$ with respect to an unknown source distribution $\mathcal{D}$. This line of work has both motivated and benefited from rich connections to the study of function complexity: underlying every new learning algorithm are new structural insights about the concept class of functions that f belongs to. Our work is motivated by the question of whether structural properties of the *source distribution* $\mathcal{D}$ can be similarly leveraged for the design of efficient algorithms.

Distribution-free learning. In Valiant's original distribution-free PAC learning model [Val84], the target function f is promised to belong to a *known*, and often "computationally simple" concept class $\mathscr{C}$, whereas no assumptions are made about the distribution $\mathcal{D}$.

Unfortunately, efficient algorithms have proven hard to come by even for simple concept classes. Consider, for instance, the class of polynomial-size DNF formulas—the motivating example that Valiant used throughout his paper. Despite decades of effort, the current fastest algorithm for this class runs in exponential time, $\exp(\tilde{O}(n^{1/3}))$ [KS04]. For other classes, even formal hardness results have been established. For example, under standard cryptographic assumptions it has been shown that there are no efficient algorithms for learning intersections of polynomially many halfspaces [KS09].

A major source of difficulty in the design of efficient learning algorithms in this model stems from the absence of assumptions about $\mathcal{D}$. Even if the concept class $\mathscr{C}$ is assumed to be simple, this can be negated by the fact that the underlying distribution $\mathcal{D}$ is arbitrarily complex. Indeed, this is precisely the idea underlying existing hardness results for distribution-free learning.

Distribution-specific learning. This leads us to the *distribution-specific* variant of PAC learning where $\mathcal{D}$ is known rather than arbitrary [BI91, Nat92]. For the domain $\{\pm 1\}^n$, the most popular assumption takes $\mathcal{D}$ to be the uniform distribution—an assumption that opens the door to many more positive results. For example, there is a quasipolynomial-time algorithm for learning polynomial-size DNF formulas under the uniform distribution [Ver90]; if the learner is additionally given query access to f, there are even polynomial-time algorithms [Jac97, GKK08]. Similarly, there is a quasipolynomial-time algorithm for learning intersections of polynomially many halfspaces under the uniform distribution [KOS04]—a sharp contrast to the distribution-free setting where even learning intersections of *two* halfspaces in subexponential time is an longstanding open problem [KS08, She13]. Indeed, there is by now a suite of powerful techniques for the design of uniform-distribution learning algorithms, notably "meta-algorithms" for learning an arbitrary concept class $\mathscr{C}$ with performance guarantees that scale with its Fourier properties. Famous examples include the algorithms for learning (approximately) low-degree functions [LMN93] and (approximately) sparse functions [KM93], as well as their agnostic variants [KKMS08, GKK08]; see Chapter §3 of [O'D14] for an in-depth treatment.

While this setting has proved to be fertile grounds for the development of sophisticated techniques and fast algorithms, it is admittedly a stylized theoretical model, since real-world data distributions often exhibit correlations and are not uniform. For this reason, uniform-distribution learners have thus far been limited in their practical relevance.

1.1 This work

These two settings, distribution-free and uniform-distribution learning, correspond to two extremes in terms of distributional assumptions. Distribution-free algorithms make no assumptions about the distribution, but the design of such algorithms that are computationally efficient has proven correspondingly challenging. On the other hand, we have a wealth of techniques and positive results in the uniform-distribution setting, but the guarantees of such algorithms only hold under a strong, often unrealistic assumption on the data distribution. Due to these vast differences, research in these two settings has mostly proceeded independently with few known connections.

The motivation for our work is the search for fruitful middle grounds between these two extremes, where efficient algorithms can be obtained for expressive concept classes and where their guarantees hold for broad classes of distributions. More generally, we ask:

> *Can we interpolate between the two extremes via notions of distribution complexity, and design learning algorithms whose efficiency scale with the inherent complexity of $\mathcal{D}$?*

Relatedly, can the large body of existing results and techniques for learning under the uniform distribution be lifted, ideally in a blackbox fashion, to non-uniform but nevertheless "simple" distributions?

2 Our results

We provide affirmative answers to these questions via a natural notion of distribution complexity:

Definition 1 (Decision tree complexity of $\mathcal{D}$)**.** *The decision tree complexity of a distribution $\mathcal{D}$ over $\{\pm 1\}^n$, denoted* $\mathrm{depth}(\mathcal{D})$*, is the smallest integer d such that its probability mass function (pmf) can be computed by a depth-d decision tree.*

Decision tree complexity interpolates between the uniform distribution on one extreme (depth 0) and arbitrary distributions on the other (depth n).

This notion, first considered by Feldman, O'Donnell, and Servedio [FOS08], generalizes other well-studied notions of complexity of high-dimensional discrete distributions and is itself a special cases of others. A *d-junta distribution* $\mathcal{D}$, introduced by Aliakbarpour, Blais, and Rubinfeld [ABR16], is one for which there exists a set of d coordinates $J \subseteq [n]$ such that for every assignment $\rho \in \{\pm 1\}^J$, the conditional distribution $\mathcal{D}_{J \leftarrow \rho}$ is uniform. Every d-junta distribution has decision tree complexity d, but a depth-d decision tree distribution can have junta complexity as large as 2^d. On the other hand, decision tree distributions are a special case of mixtures of subcubes [CM19], since a depth-d decision tree induces a partition of $\{\pm 1\}^n$ into 2^d many disjoint subcubes. Mixtures of subcubes are in turn a special case of mixtures of product distributions over discrete domains [FM99, Cry99, CGG01, FOS08].

2.1 First main result: Lifting uniform-distribution learners

Our algorithms and analyses will use the notion of *influence* of variables. This notion is most commonly applied to boolean-valued functions, and in this work we extend its study to the pmfs of distributions:

Definition 2 (Influence of variables on distributions)**.** *Let $\mathcal{D}$ be a distribution over $\{\pm 1\}^n$ and $f_{\mathcal{D}}(x) = 2^n \cdot \mathcal{D}(x)$ be its pmf scaled up by the domain size.*[1] *The influence of a coordinate $i \in [n]$ on a distribution $\mathcal{D}$ over $\{\pm 1\}^n$ is the quantity*

$$\mathrm{Inf}_i(f_{\mathcal{D}}) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^n} \left[\left| f_{\mathcal{D}}(\boldsymbol{x}) - f_{\mathcal{D}}(\boldsymbol{x}^{\sim i}) \right| \right],$$

where $\mathcal{U}^n$ denotes the uniform distribution over $\{\pm 1\}^n$ and $\boldsymbol{x}^{\sim i}$ denotes $\boldsymbol{x}$ with its i-th coordinate rerandomized. The total influence of $\mathcal{D}$ is the quantity $\mathrm{Inf}(f_{\mathcal{D}}) \coloneqq \sum_{i=1}^n \mathrm{Inf}_i(f_{\mathcal{D}})$.

With Definitions 1 and 2 in hand we can now state our first main result. For clarity, we first state it assuming a unit-time oracle that computes the influences of variables:

Theorem 1 (Lifting uniform-distribution learners; see Theorem 5 for the formal version)**.** *For any concept class $\mathscr{C}$ of functions $f : \{\pm 1\}^n \to \{0, 1\}$ closed under restrictions, assuming a unit-time influence oracle, if there is an algorithm for learning $\mathscr{C}$ under the uniform distribution to accuracy ε with sample complexity m and running time* $\mathrm{poly}(n, m)$*, there is an algorithm for learning $\mathscr{C}$ under depth-d decision tree distributions with sample complexity and running time*

$$M = \mathrm{poly}(n) \cdot \left(\frac{dm}{\varepsilon} \right)^{O(d)}.$$

As an example setting of parameters, Theorem 1 lifts a quasipolynomial-time uniform-distribution algorithm into one that still runs in quasipolynomial time, but now succeeds under any distribution with decision tree complexity $\mathrm{polylog}(n)$. We note that such distributions are quite broad: the size of a depth-d tree can be as large as $2^d = \mathrm{quasipoly}(n)$, corresponding to the mixture of that many subcubes.

The proof of Theorem 1 readily extends to lift *agnostic* uniform-distribution learners [Hau92, KSS94] to agnostic distribution-free ones. As mentioned in the introduction, many algorithms for learning under the uniform distribution are obtained through Fourier-analytic meta-algorithms. By Theorem 1, we now have analogues of these meta-algorithms for distributions with low decision tree complexity.

Estimating influences efficiently. We have stated Theorem 1 assuming that influences of variables on distributions can be computed exactly in unit time. In the body of the paper we show how these quantities can be approximated to sufficiently high accuracy given access to $\mathcal{D}$. For monotone distributions we show how this can be done using only samples from $\mathcal{D}$. For general distributions, we use the notion of *subcube conditioning samples*, proposed in [CRS15, BC18] and subsequently studied in [CCK+21, CJLW21]: in this model, the algorithm specifies a subcube of $\{\pm 1\}^n$ and receives a draw $\boldsymbol{x} \sim \mathcal{D}$ conditioned on $\boldsymbol{x}$ lying in the subcube. (For general distributions, estimating influences based only on samples is intractable as it can be easily seen to require $\Omega(\sqrt{2^n})$ many samples.)

A more general result. We obtain Theorem 1 as a corollary of a more general result which shows how every uniform-distribution learner $\mathcal{A}$ that is robust to *distributional noise* can be lifted in a way that the resulting runtime depends only on $\mathcal{A}$'s noise tolerance and not its sample complexity.

[1] This 2^n normalisation factor makes the average value of $f_{\mathcal{D}}$ exactly 1, lending to a cleaner analysis.

Theorem 1 follows since every m-sample algorithm is automatically tolerant to an $O(1/m)$ amount of distributional noise. We achieve better parameters for algorithms that are tolerant to higher amounts of noise.

2.2 Second main result: Learning decision tree distributions

Our algorithm for Theorem 1 proceeds in a two-stage manner: we first learn the decision tree structure of $\mathcal{D}$ and then use the uniform-distribution learner to learn f restricted to each of the leaves of the tree. To carry out the first stage, we give an algorithm that learns the *optimal* decision tree decomposition of a distribution $\mathcal{D}$:

Theorem 2 (Learning decision tree distributions)**.** *Let $\mathcal{D}$ be a distribution that is representable by a depth-d decision tree. There is an algorithm that returns a depth-d tree representing a distribution $\mathcal{D}'$ such that $\mathrm{dist}_{\mathrm{TV}}(\mathcal{D},\mathcal{D}') \le \varepsilon$, with high probability over the draw of samples, and with running time and sample complexity $\mathrm{poly}(n) \cdot (d/\varepsilon)^{O(d)}$. For monotone distributions, the algorithm only uses random samples from $\mathcal{D}$, and for general distributions, it uses subcube conditional samples.*

Theorem 2 is a result of independent interest in distribution learning. Even setting aside properness, the performance guarantees of our algorithm improves, quite dramatically, the prior state of the art for learning decision tree distributions and circumvents existing hardness results. Aliakbarpour, Blais, and Rubinfeld [ABR16] gave an $n^{O(k)}$ time algorithm for learning k-junta distributions, which implies an $n^{O(2^d)}$ time algorithm for learning depth-d decision tree distributions. Chen and Moitra [CM19] gave an $s^{s^3} \cdot n^{O(\log s)}$ time algorithm algorithm for learning the mixture of s subcubes, which implies a $2^{2^{O(d)}} \cdot n^d$ time algorithm for learning depth-d decision tree distributions. Finally, Feldman, O'Donnell, and Servedio [FOS08] gave an $n^{O(m^3)}$ time algorithm algorithm for learning the mixture of m product distributions, which implies an $n^{2^{O(d)}}$ time algorithm for learning depth-d decision tree distributions. These runtimes all have a doubly-exponential dependence on d, whereas ours only depends exponentially on d. Note that the runtime of any algorithm must have at least an exponential dependence on d since that is the description length of a depth-d decision tree.

None of these prior algorithms are proper, and that fact that ours is is crucial to the application to Theorem 1: the decision tree structure specifies a decomposition of $\{\pm 1\}^n$ into disjoint subcubes, and it is on these subcubes that we run our uniform-distribution learner.

Circumventing hardness results. A novel aspect of Theorem 2 is that it sidesteps existing hardness results for learning decision tree distributions. [FOS08] showed that the problem of learning depth-d decision tree distributions is as hard as that of learning depth-d decision tree *functions* under the uniform distribution. Despite significant efforts for over three decades, the current fastest algorithm for the latter problem runs in time $n^{O(d)}$ [EH89] and improving on this is a longstanding challenge of learning theory. (It contains as a special case the junta problem [BL97]—learning k-juntas in time better than $n^{O(k)}$—itself already a notorious open problem.) Theorem 2 shows that this barrier can be circumvented in two different ways: by giving the algorithm access to subcube conditional samples, and by considering monotone distributions.

Proof Overview of Theorem 2. To describe the intuition behind Theorem 2 and the role that influence plays in its proof, we begin by considering the following elementary equations:

$$\mathop{\mathbb{E}}_{\boldsymbol{b}\sim\{\pm1\}}\left[\mathrm{Inf}((f_{\mathcal{D}})_{x_i=\boldsymbol{b}})\right] = \mathrm{Inf}(f_{\mathcal{D}}) - \mathrm{Inf}_i(f_{\mathcal{D}}) \tag{1}$$

$$2\cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D},\mathcal{U}) \le \mathrm{Inf}(f_{\mathcal{D}}), \tag{2}$$

where $(f_{\mathcal{D}})_{x_i=b}$ denotes the restriction of $f_{\mathcal{D}}$ by fixing x_i to b. Equation (1), which follows from the definition of influence, says that the total influence of $f_{\mathcal{D}}$ drops by $\mathrm{Inf}_i(f)$ when restricted by x_i. Equation (2), which is a consequence of the Efron–Stein inequality, says that the total influence of a distribution upper bounds its distance to uniformity.

Together, they suggest a simple and natural algorithm for learning decision tree distributions: build a decision tree hypothesis for $\mathcal{D}$ greedily by iteratively querying the most influential variable. After sufficiently many stages, the total influence of the conditional distributions at most leaves will be close to zero (by Equation (1)), which in turn means that most leaves will be close to uniform (Equation (2)). Indeed, this intuition can be formalized using the techniques in this work to get an algorithm that learns depth-d decision tree distributions with depth-$O(d^2)$ decision tree hypotheses in time $2^{O(d^2)}$.

To obtain the improved parameters of Theorem 2, we consier a generalization of this different algorithm: instead of splitting on the single most influential variable, we consider all $O(d)$ most influential ones as candidate splits. While this involves searching over more candidates at each split, we will show that at least one of the choices leads to a high accuracy hypothesis at depth d instead of $O(d^2)$, resulting in a smaller search space of $d^{O(d)}$ instead of $2^{O(d^2)}$.

Our approach is inspired by a recent algorithm of Blanc, Lange, Qiao, and Tan for properly learning decision tree *functions* under the uniform distribution [BLQT21]. Our analysis builds on and extends theirs to the setting of unsupervised learning, which poses a number of challenges that we have to overcome. First, while highly accurate estimates of influences can be easily obtained with membership queries to the function (in our case, the pmf of $\mathcal{D}$), subcube conditioning samples provide more limited and coarse-grained information about $\mathcal{D}$. Our algorithms for estimating influences with subcube conditioning samples, and from samples alone for monotone distributions, could see further utility in other problems. Second, while it is easy to estimate how close a *function* is to a constant (relatedly, how close two functions are) via random sampling, the analogous problem of estimating the distance of *distribution* to uniformity is an intractable problem: for a distribution over $\{\pm1\}^n$, the sample complexity of estimating its distance to uniformity is $\Theta(2^n/n)$ [VV11]. There are no known improvements using subcube conditioning samples (though [BC18, CCK+21] give efficient uniformity *testers*), and the best known algorithm for monotone distributions uses $2^{n-\Theta(\sqrt{n}\log n)}$ samples [RV20]. We sidestep this barrier by showing how total influence—for which we provide efficient estimators—can be used as a good proxy for distance to uniformity.

2.3 Other related work

Conditional samples in distribution learning and testing. The subcube conditioning model falls within a recent line of work on the power of *conditional samples* in distribution learning and testing. In this more general model, which was independently introduced by Chakraborty, Fischer, Goldhirsh, and Matsliah [CFGM16] and Canonne, Ron, and Servedio [CRS15], the algorithm can specify an arbitrary subset of the domain and receive a sample conditioned on falling within this subset. Since its introduction, a large number of works have designed conditional sample algorithms,

in distribution learning and testing [Can15, FJO+15, ACK15b, SSJ17, BCG19, FLV19, CJLW21, CCK+21] and beyond [ACK15a, GTZ17, GTZ18]. Our results add to this line of work, and further reinforce the message that conditional samples (indeed, even just subcube conditional samples) can be used circumvent sample complexity lower bounds in a variety of settings.

Other access models to distributions include queries to the pmf or cdf (the *evaluation oracle* model) [BDKR05, GMV06, RS09, CR14] and giving the algorithm probability revealing samples [OS18].

Semi-supervised learning. There is extensive research in the statistical machine learning literature on leveraging unlabeled examples to improve learning. Much of this work focuses on improving sample complexity or convergence rates of existing algorithms using additional unlabelled data [GBDB+19, BDLP08]. In contrast, our work is aimed at creating *computational and sample efficient* algorithms from existing ones that are only guaranteed to work under "nice" distributions. Results of this flavour have been studied in limited ways, e.g. transforming a 1d-uniform learning algorithm to one that works on any 1d continuous distribution [BDLP08].

The work of [BOW10]. In [BOW10], Blais, O'Donnell, and Wimmer gave an algorithm for performing polynomial regression under arbitrary product distributions over $\{\pm 1\}^n$. As an application, they showed how their algorithm can be lifted to *mixtures* of product distributions via the algorithm of [FOS08] for learning mixtures of product distributions.

Like our work, this is also an example where algorithms for learning with respect to a "simple" distribution (the uniform distribution in our case and product distributions in [BOW10]'s case) can be lifted to more complex ones (decision tree distributions in our case and mixtures of product distributions in [BOW10]'s case), via a distribution learning algorithm that decomposes the more complex one into simple ones (Theorem 2 in our case and [FOS08]'s algorithm in [BOW10]'s case). The quantitative details of our transformations are incomparable: our algorithm for learning depth-d decision tree distributions run in $\mathrm{poly}(n) \cdot d^{O(d)}$ time, whereas [FOS08]'s algorithm for learning the mixture of m product distributions run in $n^{O(m^3)}$ time. (Recall that a depth-d decision tree induces a mixture of as many as $m = 2^d$ product distributions.)

3 Discussion and future work

We view our work as part of two broader and potentially fruitful approaches to PAC learning. Much of the progress in the field thus far has been guided by the design of efficient learning algorithms for successively more expressive concept classes, as measured according to various notions of function complexity; this was the approach advocated in Valiant's pioneering paper and other early works. For example, on one such axis we have small-width conjunctions as a special case of small juntas, which are in turn a special case of small-depth decision trees, which are in turn a special case of small-width DNFs, and so on, and the field seeks to design efficient algorithms for each of these classes. We believe that it is equally natural to make progress along a separate dimension, with respect to various notions of distribution complexity. The overall goal can then be cast as that of learning successively more expressive concept classes with respect to successively more expressive distributions.

Next, our algorithm is just one instantiation of a general two-stage approach to learning that is studied in the semi-supervised literature (see e.g. [BB10]): the first gathers information about

the underlying distribution, and the second exploits this distributional information to learn the target function. It would be interesting to develop more computationally efficient examples of such a two-stage approach. More broadly, there should be much to be gained from using the insights of distribution learning to counter the difficulty of distribution-free PAC learning, the crux of which is the potential nastiness of the unknown distribution.

Finally, looking beyond PAC learning, a similar gulf exists between uniform-distribution and distribution-free *testing* of function properties. The original model of property testing was defined with respect to the uniform distribution [RS96, GGR98] and much of the ensuing research has focused on this setting, with the distribution-free variant receiving increasing attention in recent years. Can uniform-distribution testers be lifted generically to the distribution-free setting? A concrete avenue towards such a result would be via a variant of our distribution decomposition lemma that runs in sublinear time.

4 Preliminaries

Notation. Given an input $x \in \{\pm 1\}^n$, coordinate $i \in [n]$, and setting $b \in \{\pm 1\}$, we use $x_{i=b}$ to refer to the input x with the i^{th} coordinate overwritten to take the value b. Similarly, given a sequence of (coordinate, value) pairs $\pi = \{(i_1, b_1), \ldots, (i_k, b_k)\}$, we use x_π to represent x with the coordinates in π overwritten/inserted with their respective values.

Given a function $f : \{\pm 1\}^n \to \mathbb{R}$, we denote the *restriction* $f_{i=b} : \{\pm 1\}^n \to \mathbb{R}$ to be the function that maps x to $f(x_{i=b})$. We define the restriction f_π analogously.

Definition 3 (Decision trees (DT))**.** *A decision trees $T : \{\pm 1\}^n \to \mathbb{R}$, is a binary tree whose internal nodes query a particular coordinate, and whose leaves are labelled by values. Each instance $x \in \{\pm 1\}^n$ follows a unique root-to-leaf path in T: at any internal node, it follows either the left or right branch depending on the value of the queried coordinate, until a leaf is reached and its value is returned.*

The set of leaves $\ell \in \text{leaves}(T)$ therefore form a partition of $\{\pm 1\}^n$, with each leaf having $2^{n-|\ell|}$ elements, where $|\ell|$ is the depth of the leaf. Every leaf ℓ also corresponds to a sequence of (coordinates, value) pairs $\pi(\ell)$ that lead to the leaf. For a function f, will sometimes use the shorthand f_ℓ to mean the restriction $f_{\pi(\ell)}$.

Definition 4 (Decision tree distribution)**.** *We say that a distribution $\mathcal{D} : \{\pm 1\}^n \to [0, 1]$ is representable by a depth-d DT, if its pmf is computable by a depth-d decision tree T. Specifically, each leaf ℓ is labelled by a value p_ℓ, so that $\mathcal{D}(x) = p_\ell$ for all $x \in \ell$. This means that the conditional distribution of all points that reach a leaf is uniform. Moreover, since $\mathcal{D}$ is a distribution, we have: $\sum_{\ell \in \text{leaves}(T)} 2^{n-|\ell|} \cdot p_\ell = 1$.*

For a given leaf ℓ, we will write $\mathcal{D}_\ell : \{\pm 1\}^{n-|\ell|} \to [0, 1]$ to represent the conditional distribution of $\mathcal{D}$ at the leaf ℓ, so that for any $x \in \{\pm 1\}^{n-|\ell|}$, we have $\mathcal{D}_\ell(x) = \mathcal{D}(x_{\pi(\ell)}) / \Pr_{y \sim \mathcal{D}}[y \in \ell]$.

We will often scale up the pmfs of our distributions by the domain size, since it makes our analysis easier. As such, we also define the weighting function:

Definition 5 (Weighting function of distribution)**.** *Let $\mathcal{E}$ be an arbitrary distribution over $\{\pm 1\}^m$. We define the weighting function:*

$$f_{\mathcal{E}}(x) \coloneqq 2^m \cdot \mathcal{E}(x).$$

Definition 6 (Monotone distribution)**.** *We furthermore say that a distribution $\mathcal{D}$ is monotone if its pmf is monotone: for $x, y \in \{\pm 1\}^n$, if $x_i \le y_i$ for all $i \in [n]$, then $\mathcal{D}(x) \le \mathcal{D}(y)$.*

Definition 7 (TV Distance)**.** *For two distributions $\mathcal{P}, \mathcal{Q}$ over a countable domain $\mathcal{X}$, we define the total variation distance:*

$$\mathrm{dist}_{\mathrm{TV}}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \sum_{x \in \mathcal{X}} |\mathcal{P}(x) - \mathcal{Q}(x)| = \frac{1}{2} \|\mathcal{P} - \mathcal{Q}\|_1.$$

Definition 8 (ℓ_1 Influence)**.** *For any function $f : \{\pm 1\}^n \to \mathbb{R}$, the influence of the i-th variable on f is given by:*

$$\mathrm{Inf}_i(f) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^n} \left[|f(\boldsymbol{x}) - f(\boldsymbol{x}^{\sim i})| \right],$$

where $\boldsymbol{x}^{\sim i}$ denotes $\boldsymbol{x}$ with the i-th coordinate re-randomised. Note that the influence of a function is defined with respect to the uniform distribution over its domain.

We further define the total influence as the sum of influences over all variables: $\mathrm{Inf}(f) \coloneqq \sum_{i=1}^n \mathrm{Inf}_i(f)$.

Fact 4.1 (Influence $\equiv$ correlation for monotone functions)**.** *Let $f : \{\pm 1\}^n \to \mathbb{R}$ be a monotone function. Then*

$$\mathrm{Inf}_i(f) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^n} [f(\boldsymbol{x}) \cdot \boldsymbol{x}_i].$$

Definition 9 (ℓ_1 Variance)**.** *For any function $f : \{\pm 1\}^n \to \mathbb{R}$,*

$$\mathrm{Var}^{(1)}(f) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{U}^n} |f(\boldsymbol{x}) - f(\boldsymbol{y})|.$$

We will also sometimes use a different definition of variance, given by the mean absolute deviation of f:

$$\mathrm{Var}_\mu(f) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^n} |f(\boldsymbol{x}) - \mathbb{E}[f]|.$$

These two definitions are equivalent, up to constant factors:

Lemma 4.2. *For a function $f : \{\pm 1\}^n \to \mathbb{R}$,*

$$\mathrm{Var}_\mu(f) \le \mathrm{Var}^{(1)}(f) \le 2\,\mathrm{Var}_\mu(f)$$

Proof. The second part follows immediately from the triangle inequality and the first is an application of Jensen's:

$$\mathrm{Var}_\mu(f) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}} \left[\left| \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{U}} [f(\boldsymbol{x}) - f(\boldsymbol{y})] \right| \right] \le \mathop{\mathbb{E}}_{\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{U}} |f(\boldsymbol{x}) - f(\boldsymbol{y})|\,. \qquad \square$$

Definition 10 (Sensitivity)**.** *For a function $f : \{\pm 1\}^n \to \mathbb{R}$ and $x \in \{\pm 1\}^n$, the sensitivity of f at x is defined to be*

$$s(f, x) = \sum_{i=1}^n \mathbb{1}[f(x) \neq f(x^{\oplus i})].$$

Furthermore, the sensitivity of f is given by its maximum sensitivity over all points:

$$s(f) = \max_{x \in \{\pm 1\}^n} \{s(f, x)\}.$$

Note that the sensitivity of a decision tree is at most the depth of the decision tree, since any point can only be sensitive to the coordinates queried on its root-to-leaf path.

4.1 Useful inequalities

We present some useful inequalities for boolean functions.

Lemma 4.3 (Efron-Stein)**.** *For any function $f : \{\pm 1\}^n \to \mathbb{R}$:*

$$\mathrm{Var}^{(1)}(f) \le \mathrm{Inf}(f).$$

Lemma 4.4 (Total influence and sensitivity)**.** *For any function $f : \{\pm 1\}^n \to \mathbb{R}$:*

$$\mathrm{Inf}(f) \le 2s(f) \cdot \mathrm{Var}^{(1)}(f).$$

Proof. Let $s = s(f)$ be the sensitivity of f and consider the set, $\mathrm{snbr}(x) = \{i \in [n] \mid f(x) \neq f(x^{\oplus i})\}$. By assumption, $|\mathrm{snbr}(x)| \le s$. We define a coupling $(\boldsymbol{x}, \boldsymbol{y}) \sim \pi$ s.t. $\boldsymbol{y}$ is often in $\mathrm{snbr}(\boldsymbol{x})$, but the marginal distributions $\pi(\boldsymbol{x})$ and $\pi(\boldsymbol{y})$ are still uniform. First, sample $\boldsymbol{x} \sim \mathcal{U}$. Then, sample $\boldsymbol{y}$ given $\boldsymbol{x}$ as follows: for each $i \in \mathrm{snbr}(\boldsymbol{x})$, let $\boldsymbol{y} = \boldsymbol{x}^{\oplus i}$ (i.e. flip the i-th coordinate of $\boldsymbol{x}$) with probability $1/s$, and with the remaining $1 - |\mathrm{snbrs}(\boldsymbol{x})|/s$ probability, take $\boldsymbol{y} = \boldsymbol{x}$. It is easy to see that the marginal distribution over $\boldsymbol{y}$ is still uniform.

Unrolling the definition of influence, we have:

$$\begin{aligned}
\mathrm{Inf}(f) &= \sum_{i=1}^{n} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} |f(\boldsymbol{x}) - f(\boldsymbol{x}^{\oplus i})| \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} \left[\sum_{i=1}^{n} |f(\boldsymbol{x}) - f(\boldsymbol{x}^{\oplus i})| \right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} \left[\sum_{i\in\mathrm{snbr}(\boldsymbol{x})} |f(\boldsymbol{x}) - f(\boldsymbol{x}^{\oplus i})| \right] && \text{(only consider nonzero terms)} \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} \left[s \cdot \sum_{i\in\mathrm{snbr}(\boldsymbol{x})} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{x}^{\oplus i})|}{s} \right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\pi} \left[s \cdot \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\pi(\cdot|\boldsymbol{x})} |f(\boldsymbol{x}) - f(\boldsymbol{y})| \right] && \text{(definition of coupling } \pi\text{)} \\
&= s \cdot \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\pi} |f(\boldsymbol{x}) - f(\boldsymbol{y})| \\
&\le s \cdot \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\pi} |f(\boldsymbol{x}) - \mathbb{E}[f]| + s \cdot \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\pi} |f(\boldsymbol{y}) - \mathbb{E}[f]| && \text{(triangle inequality)} \\
&= 2s \cdot \mathrm{Var}_{\mu}(f) && \text{(marginal distributions of } \pi \text{ are uniform)} \\
&\le 2s \cdot \mathrm{Var}^{(1)}(f). && \text{(Lemma 4.2)}
\end{aligned}$$

□

5 Our algorithmic decomposition lemma

Here we present an algorithm that constructs a decision tree of depth d for a a distribution $\mathcal{D}$, and analyze its correctness and complexity. Throughout this section, we assume access to an oracle that gives the exact influences of variables in $f_{\mathcal{D}}$ or any of its restrictions. In the next sections we

BuildDT$(\mathcal{D}, \pi, d, \tau)$:

Input: Random examples from $\mathcal{D}$, restriction π, influence oracle for $(f_{\mathcal{D}})_\pi$, depth parameter d, influence parameter τ.

Output: A decision tree T that minimizes $\mathbb{E}_{\boldsymbol{\ell} \in T}[\mathrm{Inf}((f_{\mathcal{D}})_{\boldsymbol{\ell}})]$ among all depth-d, everywhere τ-influential trees.

1. Let $S \subseteq [n]$ be the set of variables i such that $\mathrm{Inf}_i((f_{\mathcal{D}})_\pi) \geq \tau$.
2. If S is empty or $d = 0$, return the leaf labeled with $2^{|\pi|} \cdot \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ is consistent with } \pi]$.
3. Otherwise:
   (a) For each $i \in S$, let T_i be the tree such that

$$\begin{aligned} \text{root}(T_i) &= x_i \\ \text{left-subtree}(T_i) &= \textsc{BuildDT}(\mathcal{D}, \pi \cup \{x_i = -1\}, d-1, \tau) \\ \text{right-subtree}(T_i) &= \textsc{BuildDT}(\mathcal{D}, \pi \cup \{x_i = 1\}, d-1, \tau) \end{aligned}$$

   (b) Return the tree among the T_i's defined above that minimizes $\mathbb{E}_{\boldsymbol{\ell} \in T_i}[\mathrm{Inf}((f_{\mathcal{D}})_{\boldsymbol{\ell}})]$.

Figure 1: BuildDT recursively searches for the depth-d, everywhere τ-influential tree of minimal influence at the leaves.

will show that the influences can be estimated from random examples for monotone distributions, and from subcube conditional examples for general distributions.

Theorem 3 (Learning decision tree distributions)**.** *Let $\mathcal{D}$ be a distribution that is representable by a depth-d decision tree. The algorithm* BuildDT *returns a depth-d tree representing a distribution $\mathcal{D}'$ such that $\mathrm{dist}_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') \leq \varepsilon$ w.h.p.. Given access to a unit time influence oracle, its running time is $n \cdot (d/\varepsilon)^{O(d)}$.*

The algorithm BuildDT is an exhaustive search over a subset of depth-d decision trees. We characterize this subset as follows:

Definition 11 (Everywhere τ-influential)**.** *Let T be a tree and ν be an internal node with root variable $i(\nu)$. T is* everywhere τ-influential *with respect to some f if for every $\nu \in T$, we have $\mathrm{Inf}_{i(\nu)}(f_\nu) \geq \tau$.*

5.1 Correctness

Here we show that under the oracle assumptions described above, BuildDT returns a tree within TV distance ε. The proof will rely on the following fact, which relates TV distance to the uniform ℓ_1 error of the tree with respect to $f_{\mathcal{D}}$.

Fact 5.1 (TV distance = label error)**.**

$$\begin{aligned}\mathrm{dist}_{\mathrm{TV}}(\mathcal{D},\mathcal{D}') &= \frac{1}{2}\cdot\|\mathcal{D}-\mathcal{D}'\|_1\\ &= 2^{-(n+1)}\cdot\|2^n\mathcal{D}-2^n\mathcal{D}'\|_1\\ &= 2^{-(n+1)}\cdot\|f_{\mathcal{D}}-T'\|_1.\end{aligned}$$

First, we will show that BuildDT outputs a decision tree T' with small average influence at the leaves. Then, we will show that this implies that the uniform ℓ_1 error of T' with respect to $f_{\mathcal{D}}$ is small. Correctness follows from the equivalence between $2^{-(n+1)}\|f_{\mathcal{D}}-T'\|_1$ and $\mathrm{dist}_{\mathrm{TV}}(\mathcal{D},\mathcal{D}')$.

The claim that BuildDT outputs a decision tree T' with small average influence at the leaves extends a lemma from [BLQT21], instantiated here for the metric space $\mathbb{R}$ equipped with the ℓ_1-norm:

Lemma 5.2 (Theorem 5 of [BLQT21])**.** *Let $f:\{\pm1\}^n\to\mathbb{R}$ be representable by a depth-d DT T. Then there exists $T^\star$ such that the following are satisfied:*

1. *The size and depth of $T^\star$ are at most the size and depth of T,*
2. *$T^\star$ is everywhere τ-influential with respect to f,*
3. *$2^{-n}\cdot\|f-T^\star\|_1\le d\tau$.*

In our BuildDT, we cannot compute $\|f_{\mathcal{D}}-T'\|_1$ and hence cannot search for trees that minimise this error. Instead, we find trees that minimise the expected total influence at the leaves. The following lemma relates these two values:

Lemma 5.3 (Expected total influence and ℓ_1 error)**.** *Let $f:\{\pm1\}^n\to\mathbb{R}$ be representable by a depth-d DT, and let T' be any other DT. Then:*

$$\mathop{\mathbb{E}}_{\boldsymbol{\ell}\in T'}[\mathrm{Inf}(f_{\boldsymbol{\ell}})]\le 4d\cdot 2^{-n}\|f-T'\|_1$$

Proof. Since f is representable by a depth-d decision tree, its maximum sensitivity (and the sensitivity of each of its leaf restrictions) must be at most d. Therefore, for any leaf $\ell\in T'$, Lemma 4.4 asserts that $\mathrm{Inf}(f_\ell)\le 2d\cdot\mathrm{Var}^{(1)}(f_\ell)$. Moreover,

$$\begin{aligned}\mathrm{Var}^{(1)}(f_\ell) &= \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{U}^n}|f_\ell(\boldsymbol{x})-f_\ell(\boldsymbol{y})|\\ &= \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{U}^n}|f_\ell(\boldsymbol{x})-T'_\ell+T'_\ell-f_\ell(\boldsymbol{y})|\\ &\le 2\cdot\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}^n}|f_\ell(\boldsymbol{x})-T'_\ell| && \text{(Triangle ineq.)}\\ &= 2\cdot\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}^n}[|f(\boldsymbol{x})-T'(\boldsymbol{x})|\mid\boldsymbol{x}\in\ell].\end{aligned}$$

Therefore,

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{\ell}\in T'}[\mathrm{Inf}(f_{\boldsymbol{\ell}})] &\le 2d \cdot \mathop{\mathbb{E}}_{\boldsymbol{\ell}\in T'}[\mathrm{Var}^{(1)}(f_{\boldsymbol{\ell}})] \\ &\le 4d \cdot \mathop{\mathbb{E}}_{\boldsymbol{\ell}\in T'} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}^n}[|f(\boldsymbol{x}) - T'(\boldsymbol{x})| \mid x \in \ell] \\ &= 4d \cdot \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}^n} |f(\boldsymbol{x}) - T'(\boldsymbol{x})| \\ &= 4d \cdot 2^{-n} \cdot \|f(\boldsymbol{x}) - T'(\boldsymbol{x})\|_1. \end{aligned}$$

□

As a corollary of Lemma 5.2 and Lemma 5.3, we get our pruning lemma stated in terms of influences:

Corollary 5.4 (Pruning lemma with expected total influence at leaves)**.** *Let $f : \{\pm 1\}^n \to \mathbb{R}$ be representable by a depth-d DT T. Then there exists $T^\star$ such that the following are satisfied:*

1. *The size and depth of $T^\star$ are at most the size and depth of T,*
2. *$T^\star$ is everywhere τ-influential with respect to f,*
3. *$\mathbb{E}_{\boldsymbol{\ell}\in T^\star}[\mathrm{Inf}(f_{\boldsymbol{\ell}})] \le 4d^2\tau$.*

We now move to show that the tree output by BuildDT satisfies $2^{-n} \cdot \|f_{\mathcal{D}} - T'\|_1 \le \varepsilon$.

Claim 5.5. *Let $\mathcal{D}$ be a distribution that is representable by a depth-d decision tree, and let T be the output of* BuildDT*, with $\tau = \varepsilon/8d^2$. Then, with high probability, $2^{-n} \cdot \|f_{\mathcal{D}} - T\|_1 \le \varepsilon$.*

Proof. First, we claim that T minimizes $\mathbb{E}_{\boldsymbol{\ell}\in T}[\mathrm{Inf}((f_{\mathcal{D}})_{\boldsymbol{\ell}})]$ among all depth-d, everywhere τ-influential trees. This claim holds by induction on d: since

$$\mathbb{E}_{\boldsymbol{\ell}\in T}[\mathrm{Inf}((f_{\mathcal{D}})_{\boldsymbol{\ell}})] = \tfrac{1}{2}(\mathbb{E}_{\boldsymbol{\ell}\in T_{\mathrm{left}}}[\mathrm{Inf}((f_{\mathcal{D}})_{\boldsymbol{\ell}})] + \mathbb{E}_{\boldsymbol{\ell}\in T_{\mathrm{right}}}[\mathrm{Inf}((f_{\mathcal{D}})_{\boldsymbol{\ell}})]),$$

each candidate T_i minimizes influence among all depth-d, everywhere τ-influential trees with x_i at the root under the inductive assumption that T_{left} and T_{right} minimize influence for depth-$(d-1)$ trees. Then BuildDT chooses the tree of smallest influence among all the candidate T_i's, so it minimizes total influence at leaves among all trees in its search space of depth-d, τ-influential trees.

Since Corollary 5.4 establishes the existence of a tree with average total influence at leaves $\le \varepsilon/2$, it follows that the influence T's influence is also $\le \varepsilon/2$. Furthermore, we may assume by standard Hoeffding bounds that with a sample size of $\mathrm{poly}(2^d, 1/\varepsilon)$, each leaf's value estimate of $\mathbb{E}[f_{\mathcal{D}}(\boldsymbol{x}) \mid \boldsymbol{x}$ is consistent with $\ell] = 2^{|\ell|} \cdot \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x}$ is consistent with $\ell]$ is accurate to within $\pm\varepsilon/2$ w.h.p..

We can now show that $2^{-n} \cdot \|f_{\mathcal{D}} - T\|_1 \le \varepsilon$. Throughout this section, $x \in \ell$ will stand in as

shorthand for "x consistent with the restriction at ℓ".

$$\begin{aligned}
2^{-n} \cdot \|f_{\mathcal{D}} - T\|_1 &= 2^{-n} \cdot \sum_{x \in \{\pm 1\}^n} |f_{\mathcal{D}}(x) - T(x)| \\
&= 2^{-n} \cdot \sum_{\ell \in T} \sum_{x \in \ell} |(f_{\mathcal{D}})_\ell(x) - T_\ell(x)| \\
&\le 2^{-n} \cdot \sum_{\ell \in T} \sum_{x \in \ell} |(f_{\mathcal{D}})\ell(x) - \mathbb{E}[(f_{\mathcal{D}})_\ell]| + |\,\mathbb{E}[(f_{\mathcal{D}})_\ell] - T_\ell(\boldsymbol{x})| \\
&= 2^{-n} \cdot \big(\sum_{\ell \in T} 2^{n-|\ell|} \cdot \mathrm{Var}_\mu((f_{\mathcal{D}})_\ell) + \sum_{\ell \in T} 2^{n-|\ell|} \cdot |\,\mathbb{E}[(f_{\mathcal{D}})_\ell] - T_\ell|\big) \\
&\le \mathbb{E}_{\ell \in T}\, \mathrm{Inf}((f_{\mathcal{D}})_\ell)) + \mathbb{E}_{\ell \in T}\left[|\,\mathbb{E}[(f_{\mathcal{D}})_\ell(\boldsymbol{x})] - T_\ell|\right] \qquad \text{(Lemma 4.3)} \\
&\le \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \qquad \square
\end{aligned}$$

We can now prove correctness and time bounds for BuildDT.

Proof of Theorem 3. Correctness follows by combining Claim 5.5 and Fact 5.1. To bound the running time, first we note that for all restrictions π, there are at most $d/\tau = 8d^3/\varepsilon$ variables of influence at least τ. This is because all restrictions of $f_{\mathcal{D}}$ are depth-d decision trees, and for any depth-d decision tree, the sum of all variable influences is at most d.

Then the number of recursive calls to BuildDT is at most $(8d^3/\varepsilon)^d$. Each call at an internal node makes n calls to the unit-time influence oracle, and each call at a leaf processes $\mathrm{poly}(2^d, 1/\varepsilon)$ samples to estimate the leaf label. Thus, the total running time is $n \cdot (d/\varepsilon)^{O(d)}$, as desired. $\square$

6 Algorithms for computing distributional influences

In Section 5, we assumed the ability to exactly compute influences of f and its restrictions in unit time. In this section, we show how to instead estimate the influences from samples, when the distribution is monotone or when we have access to subcube conditional samples. Just as in [BLQT21], our proof only requires estimates to be accurate to $\pm \min(\tau/4, \varepsilon/n)$. Letting $\textsc{InfEst}_i(f)$ denote such an estimate of $\mathrm{Inf}_i(f)$, the pseudocode in Figure 1 and proof of Theorem 3 is modified as follows.

1. We modify S to include variables i such that $\textsc{InfEst}_i((f_{\mathcal{D}})_\pi) \ge 3\tau/4$. Since the estimate is accurate to $\pm\tau/4$, this is guaranteed to include all variables with influence $\ge \tau$, and furthermore will only include variables with influence at least $\tau/2$. Therefore, the total size of S is at most $\frac{d}{\tau/2}$, which is only a constant factor (of 2) larger than in the proof of Theorem 3 which assumed perfect influence oracles, and there does not affect asymptotic runtime or sample complexity.

2. It returns the tree T_i that minimizes $\mathbb{E}_{\boldsymbol{\ell} \in T_i}[\sum_{i \in [n]} \textsc{InfEst}_i((f_{\mathcal{D}})_{\boldsymbol{\ell}})]$. Since each estimate is accurate to ε/n, this estimate of total influence of $(f_{\mathcal{D}})_{\boldsymbol{\ell}})$ will be accurate to $\pm\varepsilon$. Then, in Claim 5.5, rather than BuildDT building a tree T minimizes $\mathbb{E}_{\boldsymbol{\ell} \in T}[\mathrm{Inf}((f_{\mathcal{D}})_{\boldsymbol{\ell}})]$ among all depth-d, everywhere τ-influential trees, T (roughly) minimizes $\mathbb{E}_{\boldsymbol{\ell} \in T}[\sum_{i \in [n]} \textsc{InfEst}_i((f_{\mathcal{D}})_{\boldsymbol{\ell}})]$ among all depth-d, everywhere τ-influential trees. More formally, T will either be the best depth-d, everywhere τ-influential trees, or better, as we know its searches over all variables

with $\textsc{InfEst}_i((f_{\mathcal{D}})_\pi) \geq 3\tau/4$ which is guaranteed to include variables with influence $\geq \tau$, but can include more variables. Finally, since the estimates of total influence are accurate to $\pm\varepsilon$, using InfEst rather than Inf can only incur at most 2ε additive error, which is a constant factor in the analysis.

Given any distribution $\mathcal{D}$ over $\{\pm 1\}^n$, we need to estimate $\mathrm{Inf}_i((f_{\mathcal{D}})_\ell)$ for any restriction ℓ of $f_{\mathcal{D}}$ and $i \in [n] - \ell$. In this section, we will instead show how to compute $\mathrm{Inf}_i(f_{\mathcal{E}})$ for any distribution $\mathcal{E}$ over $\{\pm 1\}^m$. We can then use our estimators with $\mathcal{E} = \mathcal{D}_\ell$ to obtain the necessary answers using the following fact:

Fact 6.1. *For any distribution $\mathcal{D}$ over $\{\pm 1\}^n$, restriction ℓ of $\{\pm 1\}^n$, and $i \in [n] - \ell$:*

$$\mathrm{Inf}_i((f_{\mathcal{D}})_\ell) = 2^{|\ell|} \cdot \Pr_{x \sim \mathcal{D}}[x \in \ell] \cdot \mathrm{Inf}_i(f_{\mathcal{D}_\ell}).$$

Proof. Let $w_\ell = \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \in \ell]$. We have:

$$\begin{aligned}
\mathrm{Inf}_i((f_{\mathcal{D}})_\ell) &= \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^n}\left[\left|f_{\mathcal{D}}(\boldsymbol{x}) - f_{\mathcal{D}}(\boldsymbol{x}^{\sim i})\right| \mid x \in \ell\right] \\
&= 2^n \cdot \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^n}\left[\left|\mathcal{D}(\boldsymbol{x}) - \mathcal{D}(\boldsymbol{x}^{\sim i})\right| \mid x \in \ell\right] \\
&= 2^n \cdot w_\ell \cdot \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^n}\left[\left|\frac{\mathcal{D}(\boldsymbol{x})}{w_\ell} - \frac{\mathcal{D}(\boldsymbol{x}^{\sim i})}{w_\ell}\right| \mid x \in \ell\right] \\
&= 2^n \cdot w_\ell \cdot \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{U}^{n-|\ell|}}\left|\mathcal{D}_\ell(\boldsymbol{y}) - \mathcal{D}_\ell(\boldsymbol{y}^{\sim i})\right| \\
&= 2^{|\ell|} \cdot w_\ell \cdot \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{U}^{n-|\ell|}}\left|f_{\mathcal{D}_\ell}(\boldsymbol{y}) - f_{\mathcal{D}_\ell}(\boldsymbol{y}^{\sim i})\right| \\
&= 2^{|\ell|} \cdot w_\ell \cdot \mathrm{Inf}_i(f_{\mathcal{D}_\ell}).
\end{aligned}$$

□

Because of Fact 6.1, for any restriction ℓ of depth at most d, to estimate $Inf_i((f_{\mathcal{D}})_\ell)$ to accuracy $\pm\varepsilon$, it is sufficient to estimate $\mathrm{Inf}_i(f_{\mathcal{D}_\ell})$ to accurate $\pm\varepsilon/2^d$. This 2^d factor is dominated by the $d^{O(d)}$ term in Theorem 2, so we are free to do the later.

6.1 Monotone distributions using samples

Let $\mathcal{E}$ be an arbitrary distribution over $\{\pm 1\}^m$. If $\mathcal{E}$ is monotone, the influences of $f_{\mathcal{E}}$ can be efficiently computed directly from samples of $\mathcal{E}$, via an estimate of bias:

Lemma 6.2 (Estimating influence using bias)**.** *If $\mathcal{E}$ is monotone,*

$$\mathrm{Inf}_i(f_{\mathcal{E}}) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{E}}[\boldsymbol{x}_i].$$

Proof. Using Fact 4.1,

$$\begin{aligned}
\mathrm{Inf}_i(f_{\mathcal{E}}) &= \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{U}^m}[f_{\mathcal{E}}(\boldsymbol{x}) \cdot \boldsymbol{x}_i] \\
&= \sum_{x \in \{\pm 1\}^m} 2^{-m}\ f_{\mathcal{E}}(x) \cdot x_i \\
&= \sum_{x \in \{\pm 1\}^m} \mathcal{E}(x) \cdot x_i \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{E}}[\boldsymbol{x}_i].
\end{aligned}$$

□

As a simple application of the above and Hoeffding's inequality, we obtain the following corollary.

Corollary 6.3 (Estimating influences of monotone distributions)**.** *For any $\varepsilon, \delta > 0$, there is an efficient algorithm that given unknown monotone distribution $\mathcal{E}$, computes an estimate of $\mathrm{Inf}_i(f_{\mathcal{E}})$ to accuracy $\pm\varepsilon$ with probability at least $1-\delta$ using $O(\log(1/\delta)/\varepsilon^2)$ random samples from $\mathcal{E}$.*

Recall that for Theorem 3, we only need the influence estimates to be accurate to $\pm\mathrm{poly}(2^{-d}, \tau, \varepsilon, 1/n)$. Setting $\tau = O(\varepsilon/d^2)$ and union bounding over $n \cdot (d/\varepsilon)^{O(d)}$ calls to the influence oracle gives a sample complexity of $\mathrm{poly}(n, 2^d, \varepsilon, \log(1/\delta))$. The running time is still dominated by the number of recursive calls.

Corollary 6.4 (Sample complexity of BuildDT for monotone distributions)**.** *The algorithm $\textsc{BuildDT}(\mathcal{D}, \varnothing, d, \frac{\varepsilon}{2d^2})$, given $\mathrm{poly}(n, 2^d, 1/\varepsilon, \log(1/\delta))$ random examples from a monotone distribution $\mathcal{D}$, runs in $\mathrm{poly}(n) \cdot (d/\varepsilon)^{O(d)} \cdot \log(1/\delta)$ time and outputs a distribution within TV distance ε of $\mathcal{D}$. The algorithm fails with probability at most δ.*

6.2 Beyond monotone distributions using subcube conditional sampling

We also design an influence estimator for arbitrary distributions $\mathcal{E}$ over $\{\pm1\}^m$ using subcube conditional sampling.

$\textsc{InfEst}(\mathcal{E}, i, \varepsilon)$:

Input: A distribution $\mathcal{E}$ over $\{\pm1\}^m$, coordinate $i \in [m]$, and bias parameter ε.
Output: An estimate of $\mathrm{Inf}_i(f_{\mathcal{E}})$ that has bias at most ε.

1. Sample a random $\boldsymbol{x} \sim \mathcal{E}$ and define the subcube
$$S \coloneqq \{\boldsymbol{x}\} \cup \{\boldsymbol{x}^{\oplus i}\}$$
where $x^{\oplus i}$ is x with the i^{th} bit flipped.
2. Take $\lceil 1/\varepsilon^2 \rceil$ independent samples from $\mathcal{E}$ conditioned on the output being in S, and let p be the fraction of those samples that equal $\boldsymbol{x}$.
3. Output $|p - (1-p)|$.

Figure 2: Pseudocode for estimating the influence of a variable on a distribution's weighting function.

Proposition 6.5 (InfEst has low bias)**.** *For any distribution $\mathcal{E}$ over $\{\pm1\}^m$, coordinate $i \in [m]$, and $\varepsilon > 0$,*
$$\left|\mathbb{E}\left[\textsc{InfEst}(\mathcal{E}, i, \varepsilon)\right] - \mathrm{Inf}_i(f_{\mathcal{E}})\right| \le \varepsilon$$
where InfEst is as defined in Figure 2.

Before proving Proposition 6.5, we note that it implies a high accuracy estimator.

Corollary 6.6 (Estimating influences)**.** *For any $\varepsilon, \delta > 0$, there is an efficient algorithm that given unknown distribution $\mathcal{E}$, computes an estimate of $\mathrm{Inf}_i(f_{\mathcal{E}})$ to accuracy $\pm\varepsilon$ with probability at least $1 - \delta$ using $O(\log(1/\delta)/\varepsilon^4)$ subcube conditional samples from $\mathcal{E}$.*

Proof. The algorithm outputs the mean of $O(\log(1/\delta)/\varepsilon^2)$ independent calls to $\textsc{InfEst}(\mathcal{E}, i, \varepsilon/2)$. Each call to $\textsc{InfEst}(\mathcal{E}, i, \varepsilon/2)$ gives an output bounded within $[0, 1]$. By Hoeffing's inequality, if **est** is the mean of $O(\log(1/\delta)/\varepsilon^2)$ independent calls to $\textsc{InfEst}(\mathcal{E}, i, \varepsilon/2)$,

$$\Pr\left[|\mathbf{est} - \mathbb{E}\left[\textsc{InfEst}(\mathcal{E}, i, \varepsilon/2)\right]| \geq \varepsilon/2\right] \leq \delta.$$

The result then follows from triangle inequality and Proposition 6.5. ∎

We prove that InfEst has low bias.

Proof of Proposition 6.5. For each $x \in \{\pm 1\}^n$, let

$$p(x) \coloneqq \frac{\mathcal{E}(x)}{\mathcal{E}(x) + \mathcal{E}(x^{\oplus i})}$$

be the relative weight of x in the subcube containing x and $x^{\oplus i}$. Then, we can rewrite the influence as:

$$\begin{aligned}
\mathrm{Inf}_i(f_{\mathcal{E}}) &= \sum_{x \in \{\pm 1\}^m} \frac{1}{2^m} \left| f_{\mathcal{E}}(x) - f_{\mathcal{E}}(x^{\sim i}) \right| \\
&= \sum_{x \in \{\pm 1\}^m} \left| \mathcal{E}(x) - \mathcal{E}(x^{\sim i}) \right| && (f_{\mathcal{E}}(x) = 2^m \mathcal{E}(x)) \\
&= \frac{1}{2} \cdot \sum_{x \in \{\pm 1\}^m} \left| \mathcal{E}(x) - \mathcal{E}(x^{\oplus i}) \right| && (x^{\sim i} = x^{\oplus i} \text{ wp } \tfrac{1}{2}, \text{ and otherwise } x^{\sim i} = x) \\
&= \frac{1}{2} \cdot \sum_{x \in \{\pm 1\}^m} \left(\mathcal{E}(x) + \mathcal{E}(x^{\oplus i}) \right) \cdot \left| p(x) - p(x^{\oplus i}) \right|. && (\text{definition of } p(x))
\end{aligned}$$

Then, using the fact that $p(x) = 1 - p(x^{\oplus i})$, and distributing the $(\mathcal{E}(x) + \mathcal{E}(x^{\oplus i}))$ term, we can write

$$\begin{aligned}
\mathrm{Inf}_i(f_{\mathcal{E}}) &= \frac{1}{2} \sum_{x \in \{\pm 1\}^m} \mathcal{E}(x) \cdot |p(x) - (1 - p(x))| + \mathcal{E}(x^{\oplus i}) \cdot \left| p(x^{\oplus i}) - (1 - p(x^{\oplus i})) \right| \\
&= \sum_{x \in \{\pm 1\}^m} \mathcal{E}(x) \cdot |p(x) - (1 - p(x))|
\end{aligned}$$

where, in the last step, we used the fact that summing over $x \in \{\pm 1\}^m$ is equivalent to summing over $x^{\oplus i} \in \{\pm 1\}^m$. Let $\hat{p}(x)$ be the random variable for the estimate of $p(x)$ computed by InfEst

step 2. We bound the bias of InfEst.

$$\begin{aligned}
&\left|\mathbb{E}\left[\textsc{InfEst}(\mathcal{E}, i, \varepsilon)\right] - \mathrm{Inf}_i(f_{\mathcal{E}})\right| \\
&\qquad = \left|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{E}}\left[|2\hat{p}(\boldsymbol{x}) - 1|\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{E}}\left[|2p(\boldsymbol{x}) - 1|\right]\right| && (2p-1 = p-(1-p)) \\
&\qquad = \left|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{E}}\left[|2\hat{p}(\boldsymbol{x}) - 1| - |2p(\boldsymbol{x}) - 1|\right]\right| && (\text{linearity of expectation}) \\
&\qquad \le \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{E}}\left[\big|\,|2\hat{p}(\boldsymbol{x}) - 1| - |2p(\boldsymbol{x}) - 1|\,\big|\right] && (\text{Jensen's inequality}) \\
&\qquad \le 2\left|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{E}}\left[|\hat{p}(\boldsymbol{x}) - p(\boldsymbol{x})|\right]\right|. && (\big||a| - |b|\big| \le |a-b|)
\end{aligned}$$

For any $x \in \{\pm 1\}^m$, $\hat{p}(x)$ is the average of $\lceil 1/\varepsilon^2 \rceil$ random variables each which is 1 with probability $p(x)$ and 0 otherwise. As a result, $\mathbb{E}[\hat{p}(x)] = p(x)$, and $\mathrm{Var}[\hat{p}(x)] \le \varepsilon^2/4$. Applying Jensen's inequality, we conclude

$$\left|\mathbb{E}\left[\textsc{InfEst}(\mathcal{E}, i, \varepsilon)\right] - \mathrm{Inf}_i(f_{\mathcal{E}})\right| \le 2\sqrt{\mathrm{Var}[\hat{p}(x)]} \le \varepsilon. \qquad \square$$

7 Lifting uniform distribution learners: Proof of Theorem 1

In this section, we show how to lift algorithms that learn over the uniform distribution to algorithms that learn with respect to arbitrary distributions, where the sample complexity and runtime scale with the decision tree complexity of the distribution. We first define our goal formally.

Definition 12 (Learning with respect to a class of distributions)**.** *For any concept class $\mathscr{C}$ of functions $f : \{\pm 1\}^n \to \{0,1\}$, $\varepsilon, \delta > 0$, set of distributions $\mathscr{D}$ with support $\{\pm 1\}^n$, and $m, d \in \mathbb{N}$, we say that an algorithm $\mathcal{A}$ (ε,δ)-learns $\mathscr{C}$ for distributions $\mathscr{D}$ using m samples if the following holds: For any $\mathcal{D} \in \mathscr{D}$ and any $f^\star \in \mathscr{C}$, given m iid samples of the form $(\boldsymbol{x}, f^\star(\boldsymbol{x}))$ where $\boldsymbol{x} \sim \mathcal{D}$, $\mathcal{A}$ outputs a hypothesis h satisfying*

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}}[f^\star(\boldsymbol{x}) \neq h(\boldsymbol{x})] \le \varepsilon.$$

with probability at least $1-\delta$.

Generally, we can think of δ as any fixed constant, as the success probability can always be boosted.

Fact 7.1 (Boosting success probability)**.** *Given an algorithm $\mathcal{A}$ that $(\varepsilon, \frac{1}{2})$-learns a concept $\mathscr{C}$ using m samples, for any $\delta > 0$, we can construct an $\mathcal{A}'$ that $(1.1\varepsilon, \delta)$-learns $\mathscr{C}$ using $m \cdot \mathrm{poly}(1/\varepsilon, \log(1/\delta))$ samples.*

Proof. By repeating $\mathcal{A}$ $\log(1/\delta)$ times, we can guarantee that with probability at least $1 - \delta/2$, one of the returned hypotheses is ε-close to $f^\star$. The accuracy of each of these hypothesis can be estimated to accuracy $\pm 0.05\varepsilon$ using $O(\log(1/\delta)/\varepsilon^2)$ random samples, and the most accurate one returned. With probability at least $1-\delta$, that hypothesis will have at most 1.1ε error. $\square$

Our goal is to learn with respect to the class of all low-depth decision tree distributions. We use the following natural assumption on the concept class, which includes almost every concept class considered in the learning theory literature.

Definition 13 (Closed under restriction)**.** *A concept class $\mathscr{C}$ of functions $f : \{\pm 1\}^n \to \{0,1\}$ is* closed under restriction *if, for any $f \in \mathscr{C}$, $i \in [n]$, and $b \in \{\pm 1\}$, the restriction $f_{i=b}$ is also in $\mathscr{C}$.*

We will use Theorem 3 to first decompose $\mathcal{D}$ into a mixture of nearly uniform distributions, and then run our learner on each of those distributions, as described in Figure 3.

LiftLearner$(T, \mathcal{A}, S)$:

Input: A decision tree T, an algorithm for learning in the uniform distribution $\mathcal{A}$, and a random labeled sample S.

Output: A hypothesis

For each leaf $\ell \in T$ {

1. Let S_ℓ be the subset of points in S that reach ℓ.
2. Create a set S'_ℓ consisting of points in S_ℓ but where all coordinates queried on the root-to-leaf path for ℓ are rerandomized independently (this makes the marginal over the input uniform).
3. Use $\mathcal{A}$ to learn a hypothesis, h_ℓ, with S'_ℓ as input.

}

Return the hypothesis that, when given an input x, first determines which leaf $\ell \in T$ that x follows and then outputs $h_\ell(x)$.

Figure 3: Pseudocode lifting a uniform distribution learner to one which succeeds on decision tree distributions. In this pseudocode, we assume that we have a decision tree representation which is close to the distribution, which can be accomplished using BuildDT in Figure 1.

For our first result, we will assume that we already have a learner that succeeds on distributions that are sufficiently close to uniform.

Definition 14 (Robust learners)**.** *For any concept class $\mathscr{C}$ of functions $f : \{\pm 1\}^n \to \{0,1\}$ and algorithm $\mathcal{A}$, we say that $\mathcal{A}$ (ε, δ, c)-*robustly learns *$\mathscr{C}$ using m samples under the uniform distribution if, for any $\eta > 0$ and the class of distributions*

$$\mathscr{D}_{\mathrm{TV},\eta} \coloneqq \{\textit{Distributions } \mathcal{D} \textit{ over } \{\pm 1\}^n \textit{ where } \mathrm{dist}_{\mathrm{TV}}(\mathcal{U}, \mathcal{D}) \leq \eta\},$$

$\mathcal{A}$ $(\varepsilon + c\eta, \delta)$-learns $\mathscr{C}$ for the distributions in $\mathscr{D}_{\mathrm{TV},\eta}$ using m samples.

The study of robust learners is part of a long and fruitful line of work. In particular, every learner that is robust to nasty noise [BEK02] meets our definition of robust learners.

Our result will also apply to learners that aren't explicitly robust. This is because *every* learner is robust for $c = O(m)$.

Proposition 7.2. *For any concept class $\mathscr{C}$ and algorithm $\mathcal{A}$, is $\mathcal{A}$ (ε, δ)-learns $\mathscr{C}$ using m samples under the uniform distribution, then $\mathcal{A}$ also $(\varepsilon, \delta + \frac{1}{3}, 3m)$-robustly learns $\mathscr{C}$ using m samples.*

Proof. Fix any $\eta > 0$. Our goal is to show that $\mathcal{A}$ $(\varepsilon + 3m\eta, \delta)$-learns $\mathscr{C}$ for distributions in $\mathscr{D}_{\mathrm{dist}_{\mathrm{TV}},\eta}$. If $\eta \geq \frac{1}{3m}$, this is obviously true as any hypothesis has error ≤ 1. We therefore need only consider $\eta < \frac{1}{3m}$. When $\mathcal{A}$ receives a sample from $\mathcal{U}^m$, it returns a hypothesis with error $\leq \varepsilon$ with probability at least $1 - \eta$. Instead, $\mathcal{A}$ is receiving a sample from $\mathcal{D}^m$, where $\mathrm{dist}_{\mathrm{TV}}(\mathcal{U}, \mathcal{D}) < \frac{1}{3m}$. The success probability of any test given a sample from $\mathcal{D}^m$ rather than $\mathcal{U}^m$ can only differ by at most

$$\mathrm{dist}_{\mathrm{TV}}(\mathcal{U}^m, \mathcal{D}^m) \leq m \cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{U}, \mathcal{D}) < \frac{1}{3}.$$

Therefore, for $\eta < \frac{1}{3m}$, $\mathcal{A}$ succeeds wp at least $\delta + \frac{1}{3}$, as desired. □

We now state the main result of this section.

Theorem 4. *Choose any concept class $\mathscr{C}$ of functions $\{\pm 1\}^n \to \{0, 1\}$ closed under restrictions, $\varepsilon, \delta, c > 0$, $m, d \in \mathbb{N}$, and algorithm $\mathcal{A}$ that $(\varepsilon, \delta/(2 \cdot 2^d), c)$-robustly learns $\mathscr{C}$ using m samples under the uniform distribution. For any function $f^\star \in \mathscr{C}$, distribution $\mathcal{D}$ over $\{\pm 1\}^n$, depth-d decision tree $T : \{\pm 1\}^n \to \mathbb{R}$ computing the PMF of a distribution $\mathcal{D}_T$ where*

$$\mathrm{dist}_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}_T) \leq \frac{\varepsilon}{c},$$

and sample size of

$$M = m \cdot \mathrm{poly}\left(2^d, \frac{1}{\varepsilon}, \log\left(\frac{1}{\delta}\right)\right).$$

Let $\boldsymbol{S}$ a size-M iid sample of labeled points $(\boldsymbol{x}, f^\star(\boldsymbol{x}))$ where $\boldsymbol{x} \sim \mathcal{D}$. The output of LiftLearner$(T, \mathcal{A}, \boldsymbol{S})$ *is $O(\varepsilon)$-close to $f^\star$ w.r.t $\mathcal{D}$ with probability at least $1 - \delta$.*

By Fact 7.1, an algorithm with constant failure probability could be transformed into one with the failure probability required by Theorem 4 with only a $\mathrm{poly}(d, 1/\varepsilon, \log(\delta))$ increase in the sample size. Therefore, would Theorem 4 still holds when $\mathcal{A}$ $(\varepsilon, \frac{1}{2}, c)$-robustly learns $\mathscr{C}$ if LiftLearner applies Fact 7.1 to boost the success probability of $\mathcal{A}$. Before proving Theorem 4, we show how it implies our main result.

Theorem 5 (Lifting uniform-distribution learners, formal version of Theorem 1)**.** *Choose any concept class $\mathscr{C}$ of functions $\{\pm 1\}^n \to \{0, 1\}$ closed under restrictions, $\varepsilon, c > 0$, $m, d \in \mathbb{N}$. If there is an efficient algorithm, $\mathcal{A}$, that $(\varepsilon, \frac{1}{2})$-learns $\mathscr{C}$ using m samples for the uniform distribution, then for $M = \mathrm{poly}(n) \cdot \left(\frac{dm}{\varepsilon}\right)^{O(d)}$,*

- *There is an algorithm that $(\varepsilon, \frac{1}{6})$-learns $\mathscr{C}$ using M samples for monotone distributions representable by a depth-d decision tree.*
- *There is an algorithm that uses M conditional subcube samples from $\mathcal{D}$ and M random samples labeled by the target function that learns $\mathscr{C}$ to ε-accuracy for arbitrary (not necessarily monotone) distributions representable by a depth-d decision tree.*

In both cases, the algorithm runs in time $\mathrm{poly}(n, M)$.

Proof of Theorem 5 given Theorem 4. Using Theorem 3, we can learn the input distribution to TV-distance accuracy $\frac{\varepsilon}{3m}$ with a decision tree hypothesis. By Proposition 7.2, $\mathcal{A}$ $(\varepsilon, \frac{1}{2}, 3m)$-robustly learns $\mathscr{C}$ for the uniform distribution, which can be boosted to failure probability $O(2^{-d})$ using Fact 7.1. Applying Theorem 4 gives the desired result, with the desired runtime following from Proposition 7.5. □

The remainder of this section is devoted to the proof of Theorem 4. We'll use the following proposition.

Proposition 7.3. *For any distribution $\mathcal{D}$ over $\{\pm 1\}^n$, depth-d decision tree T, $m \in \mathbb{N}$ and $p, \delta > 0$, as long as*

$$M \geq O\left(\frac{d + m + \log(1/\delta)}{p}\right) \tag{3}$$

for a random sample of M points from $\mathcal{D}$, the probability there is a leaf $\ell \in T$ satisfying:

1. *High weight: The probability a random sample from $\mathcal{D}$ reaches ℓ is at least p,*
2. *Few samples: The number of points in the size-M sample that reach ℓ is less than m*

is at most δ.

Proof. Fix a single leaf $\ell \in T$ with weight at least p. Then, the expected number of points that reach this leaf is $\mu \geq Mp$. As long as $\mu \geq 2m$, applying multiplicative Chernoff bounds,

$$\Pr[\text{Fewer than } m \text{ points reach } \ell] \leq \exp\left(-\frac{\mu}{8}\right).$$

Union bounding over all leaves 2^d, it is sufficient to choose an M where

$$2^d \exp\left(-\frac{\mu}{8}\right) \leq \delta.$$

This is satisfied for the M from Equation (3). □

We'll also use that if we have a decision tree T that has learned the PMF to $\mathcal{D}$ to high accuracy, $\mathcal{D}$ restricted to the leaves of T is, on average over the leaves, close to uniform.

Fact 7.4 (Lemma B.4 of [BLMT22])**.** *For any distribution $\mathcal{D}$ and decision tree T computing the PMF for some distribution $\mathcal{D}_T$,*

$$\sum_{\text{leaves } \ell \in T} \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D}_\ell, (\mathcal{D}_T)_\ell) \leq 2 \cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}_T).$$

Proof of Theorem 4. First, we split up the accuracy of $h = \textsc{LiftLearner}(T, \mathcal{A}, \boldsymbol{S})$ into the accuracy of the hypotheses h_ℓ learned at each leaf ℓ:

$$\Pr_{\boldsymbol{x} \sim \mathcal{D}}[h(\boldsymbol{x}) \neq f^\star(\boldsymbol{x})] = \sum_{\ell \in T} \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \cdot \Pr_{\boldsymbol{x} \sim \mathcal{D}_\ell}[h_\ell(\boldsymbol{x}) \neq f^\star(\boldsymbol{x})].$$

Each hypothesis h_ℓ is learned by running $\mathcal{A}$ on the sample $\boldsymbol{S}'_\ell$. First, we argue that for all leaves ℓ with much of $\mathcal{D}$'s weight, will, whp, have $\geq m$ samples. Applying Proposition 7.3 with probability at least $1 - \delta/2$, for all $\ell \in T$ with $\Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \geq \frac{\varepsilon}{2^d}$, the size of $\boldsymbol{S}'_\ell$ is at least m. Conditioning on that being true, we have that,

$$\begin{aligned}
\Pr_{\boldsymbol{x} \sim \mathcal{D}}[h(\boldsymbol{x}) \neq f^\star(\boldsymbol{x})] &\leq \sum_{\ell \in T} \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \cdot \Pr_{\boldsymbol{x} \sim \mathcal{D}_\ell}[h_\ell(\boldsymbol{x}) \neq f^\star(\boldsymbol{x}) \mid |\boldsymbol{S}'_\ell| \geq m] \\
&\quad + \sum_{\ell \in T} \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \cdot \mathbb{1}[\Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \leq p] \\
&\leq \sum_{\ell \in T} \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \cdot \Pr_{\boldsymbol{x} \sim \mathcal{D}_\ell}[h_\ell(\boldsymbol{x}) \neq f^\star(\boldsymbol{x}) \mid |\boldsymbol{S}'_\ell| \geq m] + \varepsilon,
\end{aligned}$$

where the last line uses that T has at most 2^d leaves and $p = \frac{\varepsilon}{2^d}$.

Each point in $\boldsymbol{S}'_\ell$ is an iid sample with the input uniform over $\{\pm 1\}^n$ and labeled by the function $f^\star_\ell$. As $\mathscr{C}$ is closed under restrictions, $f^\star_\ell \in \mathscr{C}$. Therefore,

$$\Pr_{\boldsymbol{S}}\left[\left[\Pr_{\boldsymbol{x}\sim\mathcal{D}_\ell}[h_\ell(\boldsymbol{x}) \neq f^\star(\boldsymbol{x}) \mid |\boldsymbol{S}'_\ell| \geq m]\right] \leq \varepsilon + c\cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D}_\ell, \mathcal{U})\right] \geq 1 - \frac{\delta}{2\cdot 2^d}.$$

We union bound over all 2^d leaves ℓ and the earlier event that $|\boldsymbol{S}_\ell| \geq m$ whenever $\Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell] \geq p$, we have with probability at least $1-\delta$,

$$\begin{aligned}
\Pr_{\boldsymbol{x}\sim\mathcal{D}}[h(\boldsymbol{x}) \neq f^\star(\boldsymbol{x})] &\leq \sum_{\ell\in T} \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell]\cdot(2\varepsilon + c\cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D}_\ell, \mathcal{U})) \\
&= 2\varepsilon + c\cdot \sum_{\ell\in T} \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell]\cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D}_\ell, \mathcal{U}) \\
&= 2\varepsilon + c\cdot \sum_{\ell\in T} \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x} \text{ reaches } \ell]\cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D}_\ell, (\mathcal{D}_T)_\ell) \\
&\leq 2\varepsilon + c\cdot 2\cdot \mathrm{dist}_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}_T) && \text{(Fact 7.4)} \\
&\leq 4\varepsilon. && (\mathrm{dist}_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}_T) \leq \tfrac{\varepsilon}{c})
\end{aligned}$$

As desired, we have LiftLearner learns an $O(\varepsilon)$-accurate hypothesis w.h.p. □

Proposition 7.5 (Runtime of LiftLearner)**.** *Assuming unit-time calls to $\mathcal{A}$, given a depth-d decision tree T,* LiftLearner$(T, \mathcal{A}, S)$ *runs in time $O(n2^d\cdot|S|)$.*

Proof. The number of leaves in T is at most 2^d. Since each point in S is in $\{\pm 1\}^n$, the entire sample takes $O(n\cdot|S|)$ bits to represent. For each leaf ℓ, it takes $O(n\cdot|S|)$ time to loop through this representation, find the points consistent with ℓ, and create the modified dataset S'_ℓ, and pass it into $\mathcal{A}$. Repeating this over all leaves can be done in $O(n2^d\cdot|S|)$ time. □

Remark 1 (The agnostic setting)**.** A popular variant of learning, as defined in Definition 12, is the *agnostic setting*. In this generalization of standard learning, rather than assume $f^\star \in \mathscr{C}$, $\mathcal{A}$ is required to output a hypothesis with error $\mathrm{opt}+\varepsilon$, where opt is the minimum error of a hypothesis in $\mathscr{C}$ w.r.t $f^\star$. It's straightforward to see that Theorem 4, and therefore theorem 5, extend to the agnostic setting, where if the uniform distribution learning $\mathcal{A}$ succeeds in the agnostic setting, that learner is upgraded to one that succeeds for decision tree distributions in the agnostic setting. This is because, if there is an $f\in\mathscr{C}$ with error opt w.r.t. $f^\star$, then the average error of f over the leaves of a tree T w.r.t $f^\star$ will also be at most opt.

Acknowledgements

We thank the STOC reviewers for their detailed feedback, especially for the references to the literature on semi-supervised learning.

Guy and Li-Yang are supported by NSF awards 1942123, 2211237, and 2224246. Jane is supported by NSF Award CCF-2006664. Ali is supported by a graduate fellowship award from Knight-Hennessy Scholars at Stanford University.

References

[ABR16] Maryam Aliakbarpour, Eric Blais, and Ronitt Rubinfeld. Learning and testing junta distributions. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 19–46, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. 2, 2.2

[ACK15a] Jayadev Acharya, Clément L Canonne, and Gautam Kamath. Adaptive estimation in weighted group testing. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2116–2120. IEEE, 2015. 2.3

[ACK15b] Jayadev Acharya, Clément L Canonne, and Gautam Kamath. A chasm between identity and equivalence testing with conditional queries. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, page 449, 2015. 2.3

[BB10] Maria-Florina Balcan and Avrim Blum. A discriminative model for semi-supervised learning. 57(3), 2010. 3

[BC18] Rishiraj Bhattacharyya and Sourav Chakraborty. Property testing of joint distributions using conditional samples. *ACM Transactions on Computation Theory (TOCT)*, 10(4):1–20, 2018. 2.1, 2.2

[BCG19] Eric Blais, Clément L Canonne, and Tom Gur. Distribution testing lower bounds via reductions from communication complexity. *ACM Transactions on Computation Theory (TOCT)*, 11(2):1–37, 2019. 2.3

[BDKR05] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM Journal on Computing*, 35(1):132–150, 2005. 2.3

[BDLP08] Shai Ben-David, Tyler Lu, and David Pa. Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning. In *Proceedings of the Twenty-First Annual Conference on Learning Theory*, 2008. 2.3

[BEK02] Nader H Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002. 7

[BI91] Gyora M. Benedek and Alon Itai. Learnability with respect to fixed distributions. *Theoretical Computer Science*, 86(2):377–389, 1991. 1

[BL97] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997. 2.2

[BLMT22] Guy Blanc, Jane Lange, Ali Malik, and Li-Yang Tan. Popular decision tree algorithms are provably noise tolerant. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022. 7.4

[BLQT21] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2021. 2.2, 5.1, 5.2, 6

[BOW10] Eric Blais, Ryan O'Donnell, and Karl Wimmer. Polynomial regression under arbitrary product distributions. *Machine learning*, 80(2):273–294, 2010. 2.3

[Can15] Clément L Canonne. Big data on the rise? In *International Colloquium on Automata, Languages, and Programming*, pages 294–305. Springer, 2015. 2.3

[CCK+21] Clément L Canonne, Xi Chen, Gautam Kamath, Amit Levi, and Erik Waingarten. Random restrictions of high dimensional distributions and uniformity testing with subcube conditioning. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 321–336. SIAM, 2021. 2.1, 2.2, 2.3

[CFGM16] Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. *SIAM Journal on Computing*, 45(4):1261–1296, 2016. 2.3

[CGG01] Mary Cryan, Leslie Ann Goldberg, and Paul W Goldberg. Evolutionary trees can be learned in polynomial time in the two-state general markov model. *SIAM Journal on Computing*, 31(2):375–397, 2001. 2

[CJLW21] Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. Learning and testing junta distributions with sub cube conditioning. In *Conference on Learning Theory*, pages 1060–1113. PMLR, 2021. 2.1, 2.3

[CM19] Sitan Chen and Ankur Moitra. Beyond the low-degree algorithm: mixtures of subcubes and their applications. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 869–880, 2019. 2, 2.2

[CR14] Clément Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *International Colloquium on Automata, Languages, and Programming*, pages 283–295. Springer, 2014. 2.3

[CRS15] Clément L Canonne, Dana Ron, and Rocco A Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3):540–616, 2015. 2.1, 2.3

[Cry99] Mary Cryan. *Learning and approximation Algorithms for Problems motivated by evolutionary trees*. PhD thesis, Department of Computer Science, 1999. 2

[EH89] Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989. 2.2

[FJO+15] Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Faster algorithms for testing under conditional sampling. In *Conference on Learning Theory*, pages 607–636. PMLR, 2015. 2.3

[FLV19] Eldar Fischer, Oded Lachish, and Yadu Vasudev. Improving and extending the testing of distributions for shape-restricted properties. *Algorithmica*, 81(9):3765–3802, 2019. 2.3

[FM99] Yoav Freund and Yishay Mansour. Estimating a mixture of two product distributions. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 53–62, 1999. 2

[FOS08] Jon Feldman, Ryan O'Donnell, and Rocco A Servedio. Learning mixtures of product distributions over discrete domains. *SIAM Journal on Computing*, 37(5):1536–1564, 2008. 2, 2.2, 2.2, 2.3

[GBDB+19] Christina Göpfert, Shai Ben-David, Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, and Ruth Urner. When can unlabeled data improve the learning rate? In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 1500–1518. PMLR, 25–28 Jun 2019. 2.3

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998. 3

[GKK08] Parikshit Gopalan, Adam Kalai, and Adam Klivans. Agnostically learning decision trees. In *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*, pages 527–536, 2008. 1

[GMV06] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 733–742, 2006. 2.3

[GTZ17] Themistoklis Gouleakis, Christos Tzamos, and Manolis Zampetakis. Faster sublinear algorithms using conditional sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1743–1757. SIAM, 2017. 2.3

[GTZ18] Themis Gouleakis, Christos Tzamos, and Manolis Zampetakis. Certified computation from unreliable datasets. In *Conference On Learning Theory*, pages 3271–3294. PMLR, 2018. 2.3

[Hau92] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and computation*, 100(1):78–150, 1992. 2.1

[Jac97] Jeffrey C Jackson. An efficient membership-query algorithm for learning dnf with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997. 1

[KKMS08] Adam Kalai, Adam Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008. 1

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, December 1993. 1

[KOS04] Adam R Klivans, Ryan O'Donnell, and Rocco A Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer and System Sciences*, 68(4):808–840, 2004. 1

[KS04] Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004. Special Issue on STOC 2001. 1

[KS08] Subhash Khot and Rishi Saket. On hardness of learning intersection of two halfspaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 345–354, 2008. 1

[KS09] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009. Learning Theory 2006. 1

[KSS94] Michael Kearns, Robert Schapire, and Linda Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2/3):115–141, 1994. 2.1

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993. 1

[Nat92] B. K. Natarajan. Probably approximate learning over classes of distributions. *SIAM Journal on Computing*, 21(3):438–449, 1992. 1

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 1

[OS18] Krzysztof Onak and Xiaorui Sun. Probability–revealing samples. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018. 2.3

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. 3

[RS09] Ronitt Rubinfeld and Rocco A Servedio. Testing monotone high-dimensional distributions. *Random Structures & Algorithms*, 34(1):24–44, 2009. 2.3

[RV20] Ronitt Rubinfeld and Arsen Vasilyan. Monotone Probability Distributions over the Boolean Cube Can Be Learned with Sublinear Samples. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:34, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 2.2

[She13] Alexander A Sherstov. The intersection of two halfspaces has high threshold degree. *SIAM Journal on Computing*, 42(6):2329–2374, 2013. 1

[SSJ17] Imdad S. B. Sardharwalla, Sergii Strelchuk, and Richard Jozsa. Quantum conditional query complexity. *Quantum Information & Computation*, 17(7-8):541–567, 2017. 2.3

[Val84] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 1

[Ver90] Karsten Verbeurgt. Learning dnf under the uniform distribution in quasi-polynomial time. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 314–326, 1990. 1

[VV11] Gregory Valiant and Paul Valiant. The power of linear estimators. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 403–412. IEEE, 2011. 2.2