Chapter 32

Learning the Fourier Spectrum of Probabilistic Lists and Trees

William Aiello* Milena Mihail†

Abstract

We observe that the Linial, Mansour, and Nissan method of learning boolean concepts (under uniform sampling distribution) by reconstructing their Fourier representation [LMN89] extends when the concepts are probabilistic in the sense of Kearns and Shapire [KS90].

We show that probabilistic decision lists, and more generally probabilistic decision trees with at most one occurrence of each literal, can be approximated by polynomially small Fourier representations, and that the non-negligible Fourier coefficients can be efficiently identified and estimated. Hence, all such concepts are learnable in *polynomial time* under uniform sampling distribution. This is the first instance where Fourier methods result in polynomial learning algorithms: the polynomiality of our results should be contrasted to the $n^{\text{polylog}n}$ complexities in the analogous cases of [LMN89] and [M90]. The new ingredient of our work that allows us to achieve this polynomiality is that via refined Fourier analysis we are able to isolate the polynomially small set of non-negligible Fourier coefficients that reside in a super-polynomially large area of the spectrum.

We further observe that several more general concept classes have slightly super-polynomial ($n^{\text{polylog}n}$) learning algorithms. These classes include all polynomial-size probabilistic decision trees, their convex combinations, etc. A concrete special case which results in polynomial learnability is the weighted arithmetization of k-DNF.

1 Introduction

There is a famous (and very practical) theorem by Nyquist in signal processing which roughly says the following: "If a continuous signal has bounded spectrum, then it can be completely reconstructed by discrete sampling". It is every so often the case that "uncountable versus countable" statements for infinite domains translate to "exponential versus polynomial" statements for finite domains. The approach of "learning via Fourier transforms" that appears here and elsewhere [LMN89] [M90] is, in spirit, the finite analogue of Nyquist's theorem.

In this paper we are concerned with uniform learning of probabilistic concepts.

Like boolean concepts, *probabilistic concepts* are functions with domain the n-cube. However unlike boolean concepts whose range is the set $\{0,1\}$, the range of probabilistic concepts is the interval $[0,1]$. The interpretation is as follows: For a probabilistic concept p, and for some element $\vec{x}$ of the n-cube, the label of $\vec{x}$ is 1 with probability $p(\vec{x})$ and 0 with probability $1-p(\vec{x})$. Probabilistic concepts were introduced by Kearns and Shapire [KS90] as a model that captures natural inherent uncertainties (the reader is referred to [KS90] for further justification; here we only give "weather-prediction" as the simplest of examples: "If it was raining on Monday, and it is raining on Tuesday, then it will rain on Wednesday with probability 0.8. Here $\vec{x}$ corresponds to *raining Monday and raining Tuesday*, while $p(\vec{x}) = 0.8$. However, for any specific sequence of rainy Mondays and Tuesdays, the Wednesday that follows it either rains or it does not rain").

*Bell Communications Research, Morristown NJ 07960, aiello@flash.bellcore.com.

†Bell Communications Research, Morristown NJ 07960, mihail@flash.bellcore.com.

Uniform learning is a special form of Valiant's distribution-free learning [V84] where examples of the concept to be learned are drawn according to the uniform (as opposed to arbitrary) distribution. In the uniform learning scenario there is a class of probabilistic concepts $\mathcal{P}$, one of which is the target concept p ($\mathcal{P}$ is simply a collection of concepts). The task of a learning algorithm for the class $\mathcal{P}$ is to produce a "good" approximation $\tilde{p}$ of p after a "short" training phase and some further efficient computation. During the training phase the algorithm is presented a "small" number of samples. Each sample is a uniformly generated element of the n-cube $\vec{x}$, together with a label 1 or 0 that is determined by $p(\vec{x})$.

Despite their seeming resemblance with boolean concepts, the task of learning probabilistic concepts presents significantly larger difficulties. This is reflected both in general structural theorems (for example the combinatorial dimension that determines distribution-free learnability of probabilistic concepts [KS90] is more complex than the Vapnik-Chervonenkis dimension [BEHW89]), as well as the fact that distribution-free learnability results are not known to hold for several probabilistic concepts whose boolean analogues have rather simple distribution-free learning algorithms. A prime example example of this latter situation is decision lists.

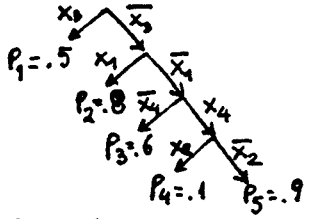

Figure 1: PROBABILISTIC DECISION LIST

For the purposes of this paper, a *probabilistic decision list* over n variables is a single branch decision tree whose edges are labeled by literals or their negations (see Figure 1). The leaves of the tree are further labeled by a number in $[0,1]$. Each element $\vec{x}$ of the n-cube naturally follows a path from the root to a unique leaf of the list. The value of the decision list on $\vec{x}$ is the label of this unique leaf.

Decision lists are generally accepted as quite natural concepts and have been studied extensively in the boolean case where the p_i's are in $\{0,1\}$ (see [R87] for a collection of results). In particular, it is well known that boolean decision lists are distribution-free learnable even in significantly more general cases [R87]. However for probabilistic decision lists the best known distribution-free result requires that the list to be learned is monotone, that is, $p_1 > p_2 > \ldots > p_{n+1}$ [KS90].

In this paper we obtain *polynomial-time uniform-learning algorithms for arbitrary probabilistic decision lists*: Theorem 3.6. Furthermore in Theorem 3.10 we show that our techniques extend to obtain polynomial-time uniform-learning algorithms for *probabilistic decision trees with a single occurrence per literal* (see Figure 2).

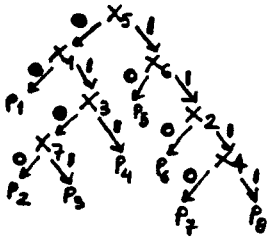

Figure 2: SINGLE LITERAR PROBABILISTIC DECISION TREE

The approach that we use for our learning purposes is to consider the Fourier representation of such concepts and approximate each Fourier coefficient. In this way, the target concept is approximated in a vector sense. This novel method of learning by approximating Fourier representations was introduced by Linial, Mansour, and Nissan in the context of boolean concepts [LMN89] (more specifically, in the context of the class AC^0 of constant depth circuits); here we make the simple observation that their techniques extend to probabilistic concepts: Theorem 2.2. The efficiency of the whole scheme depends upon the number of Fourier coefficients that must be approximated. For probabilistic decision lists and single literal probabilistic decision trees we use detailed Fourier analysis and argue that all but a polynomially small set of Fourier coefficients are negligible (Lemmas 3.2 and 3.9). We further give an algorithmic scheme that efficiently determines the frequences of the non-negligible Fourier coefficients (Remark 3.3, Algorithm 1, and discussion before Theorem 3.10). In the terms of [LMN89] and [M90], this polynomially small set of non-negligible frequences resides in the super-polynomially large area of "low frequences". Both conceptually and technically, the part of our work that filters the polynomially small set of significant frequences from a super-polynomially large area of low frequences is entirely new. This is also the reason why we obtain polynomial results instead of the $n^{\text{polylog}n}$ complexities of [MLN89] and [M90]

in analogous cases. We consider these polynomial results as the main thrust of our work.

For arbitrary probabilistic decision trees of polynomial size (i.e. when the single literal condition is removed) we observe that $n^{\text{polylog}n}$ uniform learning is feasible: Theorem 4.1. This observation follows along the line of reasoning in [LMN89], and with the further simplification that the use of Hastad's Switching Lemma [H86] is unecessary ([LMN89] make crucial use of this Lemma in their AC^0 proof). We also observe that all convex combinations concepts that have negligible high frequences preserve the low-frequency property, and are hence also learnable in the uniform model: Theorem 4.2. An interesting consequence of this observation is that the weighted arithmetization of k-DNF is learnable in polynomial time in the uniform model, and by the recent results in [M90] also distribution-free! (Theorems 4.3 and 4.4)

With respect to the potential of all these results to extend to distribution-free learning, we should point out that Mansour has developed some fairly elegant techniques to transform Fourier methods for uniform learning to distribution-free learning [M90]. Mansour's techniques apply to concepts whose Fourier representations are exactly zero on high frequences (as opposed to approximately zero). Neither AC^0 nor the probabilistic classes that are described here satisfy this requirement. However on the positive side, Mansour's work suggests that Fourier methods for uniform learning could be a first approach for the general case of distribution-free learning.

Finally, before proceeding to the technical presentation of our work, it is worth mentioning that the all Fourier methods for uniform learning that appear here and elsewhere [LMN89] [M90], except for their remarkable comparability to Nyquist's theorem, posses a variety of further desirable algorithmic features: they are conceptually simple, very easy to implement, parallelizable etc.

The rest of this extended abstract is organized as follows: In Section 2 we establish the context and technical background of our work. In Section 3 we present the polynomial learnability results for decision lists and single literal decision trees. In Section 4 we discuss arbitrary decision trees of polynomial size and convex combinations of concepts with bounded spectrum. Summary and open problems are in Section 5.

2 Preliminaries

In this section we briefly review the Fourier approach to learning. The only new point is equation (2) which suggests Theorem 2.2, namely, that the techniques in [LMN89] extend for probabilistic concepts.

- The set of *objects* is $Q_n = \{0,1\}^n$. Elements of Q_n are denoted by n-bit vectors $\vec{x} = x_1 \ldots x_n$.
- A *n-concept* (or simply a *concept* when n is well understood) is a function from the n-cube Q_n to the unit interval, $p : Q_n \to [0,1]$.
- $\mathcal{P}_n$ is a set of n-concepts. $\mathcal{P} = \cup_n \mathcal{P}_n$ is a *concept class.*
- A *sample of a concept p* is a pair $(\vec{x}, l_{\vec{x}})$, where $\vec{x}$ is drawn uniformly from Q_n, and $l_{\vec{x}}$ is 1 with probability $p(\vec{x})$ and 0 with probability $1 - p(\vec{x})$.
- A function $\tilde{p}$ is an *ϵ-approximation* of p if $(1/2^n)\sum_{\vec{x}} |p(\vec{x}) - \tilde{p}(\vec{x})| \leq \epsilon$.

Definition 2.1 [Analogous to [KS90]] *A concept class $\mathcal{P}$ is* learnable in the uniform model and in time-complexity $f(n, \epsilon, \delta)$, *if there is an algorithm that for any n and any $p \in \mathcal{P}_n$ takes as input ϵ, δ and m independent samples of p and produces with probability $1 - \delta$ a hypothesis $\tilde{p}$ which is an ϵ-approximation of p; moreover the running-time is $f(n, \epsilon, \delta)$. We say that $\mathcal{P}$ is learnable in* polynomial time *if $f(n, \epsilon, \delta)$ is polynomial in n, ϵ^{-1}, and $\log \delta^{-1}$. We say that $\mathcal{P}$ is learnable in* slightly super-polynomial time *if $f(n, \epsilon, \delta)$ is of the form $(n\epsilon^{-1})^{\text{polylog}n\epsilon^{-1}}$ and polynomial in $\log \delta^{-1}$.*

Concepts can be viewed as elements of the 2^n-dimensional vector space of all real valued functions on the n-cube (one dimension for each domain-point). In this context, determining concepts by their values on each vertex of the n-cube is equivalent to using the standard basis. Of course the obvious difficulty of the learning task is that all directions of the standard basis are equally important, and we are required to correctly approximate the projection of p on all but a vanishingly small fraction of these directions after seeing the behavior of p on a vanishingly small fraction of the directions (i.e. the small set of samples). In an effort to overcome this fundamental difficulty, Linial, Mansour, and Nissan [LMN89] introduced the idea of switching bases from the standard basis to a Fourier basis, so that the projection of p along most directions of the Fourier basis is negligible. Hence the number of non-negligible directions is comparable to the small number of samples.

The *Fourier basis* for the set of all real valed functions on the n-cube is defined as follows: For each

$S \subset [n]$ consider a "parity" function associated with S in the natural way:

$$\chi_S(\vec{x}) = (-1)^{par_S(\vec{x})}$$

where $par_S(\vec{x}) = 0$ if $\sum_{i \in S} x_i$ is even, and $par_S(\vec{x}) = 1$ if $\sum_{i \in S} x_i$ is odd. It is well known and easy to verify that $\cup_S \{\chi_S\}$ are indeed an orthonormal set with respect to the inner product $\langle p, q \rangle = \frac{1}{2^n} \sum_{\vec{x}} p(\vec{x}) q(\vec{x})$. Hence, any real valued function on the n-cube (therefore any concept p) can be written as

$$p(\vec{x}) = \sum_S \langle p, \chi_S \rangle \chi_S(\vec{x}).$$

The Sth Fourier coefficient of p is

$$\begin{aligned} a(S) &= \langle p, \chi_S \rangle \\ &= \frac{1}{2^n} \sum_{\vec{x}} p(\vec{x}) (-1)^{par_S(\vec{x})} \qquad (1) \\ &= E\left(l_{\vec{x}} (-1)^{par_S(\vec{x})} \right) \\ &= E\left(\frac{1}{m} \sum_i l_{\vec{y}_i} (-1)^{par_S(\vec{y}_i)} \right) \qquad (2) \end{aligned}$$

where $(\vec{x}, l_{\vec{x}})$ and the $(\vec{y}_i, l_{\vec{y}_i})$'s are independent samples of p. The learning algorithms that follow are crucially based on the fact that *Fourier coefficients are simply averages*, as suggested by (1), and that these averages can be efficiently estimated, as suggested by (2).

In particular, say that a concept class $\mathcal{P}$ has *bounded spectrum*, if for all concepts $p \in \mathcal{P}$ all "high" frequency Fourier coefficients (the natural notion of frequency for χ_S and $a(S)$ is the cardinality of S) are negligible in the following sense:

$$\begin{aligned} \textstyle\sum_{S:|S|>k} a^2(S) &\leq \text{poly}(n) 2^{-k^c} \\ \textstyle\sum_{S:|S|>k} a^2(S) &\leq \epsilon^2/2 \end{aligned}$$

where c is a constant and $k = \text{poly}(\log n, \log \epsilon^{-1})$. Then we may approximate the Fourier representation of p along the following lines: First use $\tilde{a}(S) = 0$ for all S: $|S| > k$. Then use m independent samples $(\vec{y}_1, l_{\vec{y}_1}), \ldots, (\vec{y}_m, l_{\vec{y}_m})$ and compute $\tilde{a}(S) = (1/m) \sum_{i=1}^m l_{\vec{y}_i} (-1)^{par_S(\vec{y}_i)}$ for all S: $|S| \leq k$. If m is some appropriate polynomial in $\binom{n}{k}$ and $\log \delta^{-1}$, then straightforward Chernoff bounds guarantee that, with probability at least $1-\delta$, $|a(S) - \tilde{a}(S)|^2 \leq k^{-1} \binom{n}{k}^{-1} \epsilon^2/2$ for each S. Hence

$$\sum_{S:|S| \leq k} (a(S) - \tilde{a}(S))^2 \leq \epsilon^2/2$$

Now if we use as a hypothesis for p the concept $\tilde{p}$ whose Fourier coefficients are the $\tilde{a}$'s, then we have:

$$\begin{aligned} \tfrac{1}{2^n} \textstyle\sum_{\vec{x}} |p(\vec{x}) - \tilde{p}(\vec{x})| &= \sqrt{\left(\tfrac{1}{2^n} \sum_{\vec{x}} |p(\vec{x}) - \tilde{p}(\vec{x})| \right)^2} \\ &\leq \sqrt{\tfrac{1}{2^n} \textstyle\sum_{\vec{x}} (p(\vec{x}) - \tilde{p}(\vec{x}))^2} \\ &= \sqrt{\textstyle\sum_S (a(S) - \tilde{a}(S))^2} \end{aligned} \qquad (3)$$

where the last two lines follow by Cauchy-Schwartz and Parseval respectively. Now this last quanity is bounded by ϵ by the previous remarks, and hence $\tilde{p}$ is a an ϵ-approximation for p with high probability. (The technicality that for some $\vec{x}$'s we may have $\tilde{p}(\vec{x}) < 0$ or $\tilde{p}(\vec{x}) > 1$ can be trivially bypassed by setting these values to 0 or 1 respectively; notice that this only helps the errors in (3)).

All the above imply:

Theorem 2.2 [Extension of [LMN90]] *If a probabilistic concept class $\mathcal{P}$ has bounded spectrum, then $\mathcal{P}$ is learnable in the uniform model and in slightly super-polynomial time.*

In [LMN89] it was shown that the class AC^0 has bounded spectrum and is hence learnable in slightly super-polynomial time. The intuition behind their result is as follows: "The spectrum of the class AC^0 concentrates on the low frequences because it should differ significantly from the spectrum of the parity function which is known not to be in AC^0 [Y85] [H86], and whose spectrum consists of a single frequency: the highest." The results that we observe in Section 4 with respect to polynomial-size probabilistic decision trees (as well as the stronger polynomial results of Section 3) are justified by the same intuition: "The spectrum of poly-size decision trees concentrates on the low frequences because it should differ significantly from the spectrum of parity which requires exponential size decision trees."

3 Probabilistic Decision Lists and Single Literal Decision Trees

In the previous section we sketched how bounded spectrum concept classes can be learned in the uniform model and in in slightly super-polynomial time. In fact, without using additional structure, this is probably the best possible since roughly $\binom{n}{\log n}$ Fourier coefficients must be approximated. The main contribution of this paper is to show that for lists and single literal decision trees there

is a polynomial size subset of the lowest $\binom{n}{\log n}$ frequences in which most of the power is concentrated, and that furthermore, this subset can be efficiently identified. Thus we obtain the first nontrivial example of a concept class with bounded spectrum for which it is possible to take advantage of further properties of the spectrum and get a polynomial time uniform learning algorithm.

The case of lists is treated in detail. The case of decision trees follows along the same general lines and is simply sketched.

Before presenting the learning algorithms let us formalize some useful terminology.

As mentioned informally in the introduction, a decision list on n variables is a single-branch binary tree with edges labeled by literals x_i and their negations $\bar{x}_i$, so that if the right edge of an internal node is labeled x_i, (resp. $(\bar{x}_i)$) then the left edge is labeled $\bar{x}_i$, (resp. (x_i)). There are $n+1$ leaves labeled by $p_1 \dots p_{n+1}$. Any $\vec{x} \in Q_n$ naturally follows a path in the tree from the root to a unique leaf. The value of a decision tree on $x \in Q_n$ is the label of this unique leaf.

- To formalize the order in which the literals appear along the decision list we introduce a permutation π of $[n]$. For example, in Figure 1 we have $\pi(1)=3$, $\pi(2)=1$, $\pi(3)=4$, $\pi(4)=2$. In general the ith level edges of the list are labeled $x_{\pi(i)}$ and $\bar{x}_{\pi(i)}$. Conversely, x_i and $\bar{x}_i$ are on level $\pi^{-1}(i)$ of the list. For convenience define the level function $l(i) = \pi^{-1}(i)$.
- To denote whether $x_{\pi(i)}$ or $\bar{x}_{\pi(i)}$ label the left edge of the i-th branch of the list, we introduce an n-bit 0-1 vector $\vec{d}$. For example, in Figure 1 we have $\vec{d} = 1101$ since the left edges are labeled by x_3, x_1, $\bar{x}_4$, x_2. We further use the notation $\bar{d}_i$ to denote the complement of d_i: $\bar{d}_i = 1 - d_i$.
- To denote the partition of the n-cube which is naturally suggested by the branches of the decision list we introduce:

$$C_i = \{\vec{x} : \begin{array}{l} x_{\pi(1)} = \bar{d}_1, \dots, x_{\pi(i-1)} = \bar{d}_{i-1}, \\ x_{\pi(i)} = d_i\}, \text{ for } 1 \le i \le n \end{array}$$

$$C_{n+1} = \{\bar{d}_1 \dots \bar{d}_n\}$$

Clearly, $|C_i| = 2^{n-i}$ for $1 \le i \le n$, and of course $|C_{n+1}| = 1$.

In the above terms, a decision list is formally defined as follows:

Definition 3.1 *A concept p over n variables is a* decision list *if for some permutation π of $[n]$, some n-bit 0-1 vector $\vec{d}$, some $p_1, \dots, p_{n+1} \in [0,1]$, and all $\vec{x} = x_1 \dots x_n \in Q_n$ the following holds: $\vec{x} \in C_i \Rightarrow p(\vec{x}) = p_i$, where the C_i's are as previously defined.*

For a subset of variables $\{x_i : i \in S \subset [n]\}$ it turns out crucial to identify the variable which is farthest down the list. To do this we define the maximum level of S or $MaxL(S)$ to be $j \in S$: $l(j) \ge l(i)$ for all $i \in S$. Now $x_{MaxL(S)}$ is the desired variable. In turn, this suggests a partition of all the subsets of the variables according to the maximum level variable in the set:

$$\emptyset, \{S : MaxL(S) = 1\}, \dots, \{S : MaxL(S) = n\}.$$

Clearly $|\{S : MaxL(S) = i\}| = 2^{i-1}$, so that the cardinalities of partition classes are polynomial for $i = O(\log n)$.

The lines along which we shall efficiently approximate the spectrum of decision lists are, roughly, the following:

(i) $\sum_{S:MaxL(S)>i} a^2(S) < 2^{-i}$, which suggests that *for the purpose of approximating the Fourier representation it suffices to approximate each one of $a(S)$ such that $MaxL(S) \le i$, for $i = O(\log n)$.* Now the set $\{S : MaxL(S) \le i\}$ is the powerset of $\{\pi(1), \dots, \pi(i)\}$ so *there are only polynomially many coefficients to approximate*, for $i = O(\log n)$. *Each one of these coefficients can be approximated efficiently and satisfactory in polynomial time* by sampling.

(ii) *The function π on $1, \dots, i$ can be approximated satisfactorily in polynomial time* by further sampling.

Point **(i)** is justified by Lemma 3.2 below which captures all the structure of the spectrum. Point **(ii)** is justified by Remark 3.3 and the learning Algorithm 1 that follows.

Lemma 3.2 *For all $S \subseteq [n]$, if $MaxL(S) = i$ then*

$$|a(S)| = \frac{1}{2^n}|2^{n-i}p_i - \sum_{j=i+1}^{n} 2^{n-j}p_j - p_{n+1}| \quad (4)$$

$$|a(S)| = |a(\{\pi(i)\})| \le 2^{-i} \quad (5)$$

$$\sum_{S:MaxL(S)>i} a^2(S) < 2^{-i} \quad (6)$$

PROOF. To see (4) recall by (1) that

$$\begin{aligned} |a(S)| &= \frac{1}{2^n}|\sum_{\vec{x}} p(\vec{x})(-1)^{par_S(\vec{x})}| \\ &= \frac{1}{2^n}|\sum_{j=1}^{n} p_j \sum_{\vec{x} \in C_j} (-1)^{par_S(\vec{x})}| \\ &= \frac{1}{2^n}|\sum_{j=i}^{n} p_j \sum_{\vec{x} \in C_j} (-1)^{par_S(\vec{x})}| \\ &= \frac{1}{2^n}|2^{n-i}p_i - \sum_{j=i+1}^{n} 2^{n-j}p_j - p_{n+1}| \end{aligned}$$

where the last two equalities are fairly easy to justify as follows:

• Clearly $S \subseteq \{\pi(1), \ldots, \pi(i)\}$. For each j such that $j < i$, we argue that $\sum_{\vec{x} \in C_j} (-1)^{par_S(\vec{x})} = 0$. This is because all the vectors in C_j have coordinates $x_{\pi(1)}, \ldots, x_{\pi(j-1)}, x_{\pi(j)}$ forced to $\bar{d}_1, \ldots, \bar{d}_{j-1}, d_j$ respecively, while each one of the coordinates $x_{\pi(j+1)} \ldots x_{\pi(i)}$ is free to vary in $\{0,1\}$. Consequently, for all such vectors the parity of the sum of their bits that belong to $S \cap \{x_{\pi(1)} \ldots x_{\pi(j)}\}$ is fixed. And on the other hand, the parity of the sum of their bits that belong to $S \cap \{x_{\pi(j+1)} \ldots x_{\pi(i)}\}$ is half the times even and the other half odd. Hence, the quantity $(-1)^{par_S(\vec{x})}$ averaged over all vectors in C_j is zero.

• For $j = i$, there are 2^{n-i} vectors in C_i. Moreover, all vectors in C_i have coordinates $x_{\pi(1)}, \ldots, x_{\pi(i-1)}$, $x_{\pi(i)}$ forced to $\bar{d}_1, \ldots, \bar{d}_{i-1}, d_i$ respectively. Hence $par_S\vec{x}$ is fixed. Similarly, for $i < j \leq n$, there are 2^{n-j} vectors in C_j. Furthermore all vectors in C_j have coordinates $x_{\pi(1)}, \ldots, x_{\pi(i)}$ forced to $\bar{d}_1, \ldots, \bar{d}_i$ respectively. Hence, $par_S(\vec{x})$ for $\vec{x} \in C_j$, $i < j \leq n$, is the complement of $par_S(\vec{x})$ for $\vec{x} \in C_i$. The case when $j = n+1$ is easily seen to be p_{n+1} with the sign as given in (4).

Equation (5) follows from equation (4) by noticing that all the p_i's are in $[0,1]$.

To verify (6) recall that there are 2^{j-1} elements in $\{S : MaxL(S) = j\}$, and by (5) each of them is at most 2^{-j}.

□

Remark 3.3

As we discussed before, in principle, for our learning purposes we would wish to know the set $X(i) = \{\pi(1), \pi(2), \ldots, \pi(i)\}$ for, say , $i = O(\log n)$. If it was the case that $|a(\{\pi(i)\})| = 2^{-i}$, then $X(i)$ could be identified as follows:

(a) For each $\pi(j) \in X(i)$ we would have $|a(\{\pi(i), \pi(j)\})| = 2^{-i}$ which follows from (5).

(b) For each $\pi(j_1) \notin X(i)$ (hence $j_1 > i$) and for all j_2, we would have $|a(\{\pi(j_1), \pi(j_2)\})| \leq 2^{-(i+1)}$ by (5) since $MaxL(\{\pi(j_1), \pi(j_2)\}) > i$.

Hence if it was the case that $|a(\{\pi(i)\})| = 2^{-i}$, then we would be able to use a small sample, approximate with high probability all $a(\{\pi(j)\})$'s and $a(\{\pi(j_1), \pi(j_2)\})$'s up to arbitrary accuracy, and by **(a)** and **(b)** above this would suffice to isolate $X(i)$.

However, since the condition $|a(\{\pi(i)\})| = 2^{-i}$ is in general not true, we need some further technicalities. In particular, let i_0 be $\max\{j : a(\{\pi(j)\}) \geq 2^{-i}\}$. Clearly, $i_0 \leq i$ by (5). Let further $X_0(i) = \{\pi(1), \pi(2), \ldots, \pi(i_0)\}$. Clearly $X_0(i) \subseteq X(i)$. The algorithm that follows uses the idea of computing $a(S)$ for $|S| = 1$ and $|S| = 2$ that was described in **(a)** and **(b)**, and isolates some X^* : $X_0(i) \subseteq X^* \subseteq X(i)$. We will argue that this suffices for learning purposes.

ALGORITHM 1: Learns Decision Lists

BEGIN

Stage 1: Approximate $X_0(i)$ and $X(i)$ by X^*.

Set $m := 16n^2\epsilon^{-4}\left(\log 2(n+1) + \log \delta^{-1}\right)$;

Set $i := \log 2\epsilon^{-2} + \log n$;

Input m samples $(\vec{y}_1, l_{\vec{y}_1}), \ldots, (\vec{y}_m, l_{\vec{y}_m})$;

$X^* := \emptyset$;

For j_1:=1 to n do $\tilde{a}(\{j_1\}) := \frac{1}{m}\sum_t l_{\vec{y}_t}(-1)^{par_{\{j_1\}}(\vec{y}_t)}$;

For $j_2 := 1$ to n, such that $j_2 \neq j_1$ do

$\tilde{a}(\{j_1, j_2\}) := \frac{1}{m}\sum_t l_{\vec{y}_t}(-1)^{par_{\{j_1, j_2\}}(\vec{y}_t)}$;

If $\tilde{a}(\{j_1\}) \geq \frac{3}{4}2^{-i}$ or for some j_2 $\tilde{a}(\{j_1, j_2\}) \geq \frac{3}{4}2^{-i}$ then $X^* := X^* \cup \{j_1\}$;

Stage 2: Approximating the Spectrum.

Set $m := n\epsilon^{-6}\left(\log 8n + \log \delta^{-1} + \log \epsilon^{-2}\right)$;

Input m samples $(\vec{y}_1, l_{\vec{y}_1}), \ldots, (\vec{y}_m, l_{\vec{y}_m})$;

For each $S \subseteq X^*$, $\tilde{a}(S) := \frac{1}{m}\sum_t l_{\vec{y}_t}(-1)^{par_S(\vec{y}_t)}$;

END.

The hypothesis is $\tilde{p}$: $\tilde{p}(\vec{x}) = \sum_{S \subseteq X^*} \tilde{a}(S)(-1)^{par_S(\vec{x})}$;

(If $\tilde{p}(\vec{x}) < 0$, then $\tilde{p}(\vec{x}) := 0$; If $\tilde{p}(\vec{x}) > 1$, then $\tilde{p}(\vec{x}) := 1$;)

Claims 3.4 and 3.5 below justify the correctness of Algorithm 1.

Claim 3.4 *At the end of Stage 1, $X_0(i) \subseteq X^* \subseteq X(i)$ with probability at least $1 - \delta/2$.*

PROOF (sketch). Follows in the spirit of Remark 3.3 and standard Chernoff bounds.

Claim 3.5 *At the end of Stage 2, $\frac{1}{2^n}\sum_{\vec{x}} |p(\vec{x}) - \tilde{p}(\vec{x})| \leq \epsilon$ with probability at least $1 - \delta/2$.*

PROOF (sketch). Assume that at the end of Stage 1 X^* is as in Claim 3.4, so that there are at most 2^i $\tilde{a}(S)$'s to be approximated. Then standard Chernoff bounds suggest that for the particular choice of m the sum of the squares of all these $\tilde{a}(S)$'s is bounded by $\epsilon^2/2$ with the desired probability. Hence:

$$\begin{aligned}
&\frac{1}{2^n}\sum_{\vec{x}} |p(\vec{x}) - \tilde{p}(\vec{x})| \leq \sqrt{\textstyle\sum_S \left(a(S) - \tilde{a}(S)\right)^2}, \text{ by (3)} \\
&= \sqrt{\textstyle\sum_{S \subseteq X^*} \left(a(S) - \tilde{a}(S)\right)^2 + \sum_{S \not\subseteq X^*} a^2(S)} \\
&\leq \sqrt{\epsilon^2/2 + \textstyle\sum_{S \not\subseteq X(i_0)} a^2(S)} \\
&\leq \epsilon
\end{aligned}$$

Where the last bound holds because:

$$\sum_{S \not\subseteq X(i_0)} a^2(S) = \sum_{S:MaxL(S)>i_0} a^2(S)$$
$$= \begin{cases} \sum_{j=i_0+1}^{i} \left(\sum_{S:MaxL(S)=j} a^2(S)\right) \\ + \sum_{S:MaxL(S)>i_0} a^2(S) \end{cases}$$
$$\leq \begin{cases} \sum_{j=i_0+1}^{i} \left(\sum_{S:MaxL(S)=j} 2^{-2i}\right) + 2^{-i} \\ \text{by (5), (6), and} \\ \text{the definition of } i_0 \end{cases}$$
$$\leq \sum_{j=i_0+1}^{i} 2^j 2^{-2i} + 2^{-i} \leq n2^{-i}$$
$$\leq \epsilon^2/2, \text{ by choice of } i \text{ in Algorithm 1}$$

□

All the above imply:

Theorem 3.6 *The class of probabilistic decision lists is learnable in polynomial time and in the uniform model via Algorithm 1.*

In the rest of this section we sketch the structure and learning algorithm for decision trees with a single occurrence per literal. The idea here is completely analogous to the case of decision lists (see Figure 3).

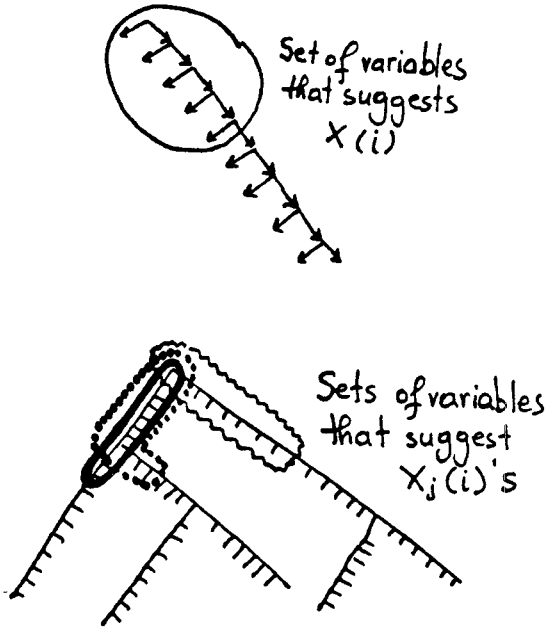

Figure 3

A *probabilistic decision tree with single occurrence per literal* over n variables can be described as follows: Let T be a binary tree with at most n interior nodes. Consider a labeling of the nodes of T with the variables $x_1, \ldots, x_n$, so that each variable labels at most one node (see Figure 2). Consider a labeling of the leaves of T with numbers in [0,1]. Each element $\vec{x}$ of the n-cube follows a path of the tree from the root to a unique leaf. The value of the probabilistic decision tree on $\vec{x}$ is the label of this unique leaf.

Analogously to the permutation π of decision lists (which can be also viewed as a total order on $[n]$: $\pi(i) < \pi(i+1)$), a decision tree defines a partial order σ on $[n]$ in the natural way: If x_i labels a descendent of x_j, then $i >_\sigma j$. Again analogously to decision lists, for some $S \subseteq [n]$, if all elements in S are related in σ (hence all elements in S appear on some path from the root to a leaf) then define $Max(S)$ to be the largest element in S. The structural Lemmas 3.7, 3.8, and 3.9 that follow are the analogues of Lemma 3.2 for decision lists, and can be shown by similar manipulations (the proofs are left for the complete paper):

Lemma 3.7 *For all $S \subseteq [n]$, if there are at least two elements in S that are not related in σ, then $a(S) = 0$.*

Lemma 3.8 *For all $S \subseteq [n]$ such that all elements in S are related in σ, let $i := Max(S)$. Then $a(S) = a(\{Max(S)\}) \leq 2^{-i}$.*

Let $Y_1, \ldots, Y_k$ be the sets of variables that appear along each one of the k paths from the root to each leaf ($k \leq n$). Let $X_1(i), \ldots, X_k(i)$ be the subsets of the Y_j's that correspond to the i smallest variables of each Y_j, that is, the first i nodes that are encountered along each path (and if the path is shorter than i then the corresponding $X_j(i)$ equals to the corresponding Y_j).

Lemma 3.9 $\sum_{S:S \not\subseteq X_j(i)} a^2(S) \leq n2^{-i}$.

So to approximate the spectrum of the tree, it suffices to approximate $a(S)$ for $S \subseteq X_j(i)$ for all j. There are at most n such $X_j(i)$'s and the powerset of each one of them, for say $i = O(\log n)$, is polynomially small. Hence there is a polynomially small number of coefficients to be approximated.

Furthermore, the crucial sets $X_j(i)$ can be approximately isolated as in Stage 1 of Algorithm 1 by estimating $a(S)$ for $|S| = 1$ and $|S| = 2$, roughly, as follows:

- For each $j_1 \in [n]$,
  let $X^*_{j_1}(i) = \{j_1\} \cup \{j_2 : \tilde{a}(\{j_1, j_2\}) \geq \frac{3}{4}2^{-i}\}$;
- If $|X^*_{j_1}(i)| > c \log n$ then $X^*_{j_1}(i) := \emptyset$;

• The sets $X_j(i)$ are approximated by the sets $X^*_{j_1}(i)$;

All this discussion can be formalized to Theorem 3.10 which concludes the section.

Theorem 3.10 *The class of probabilistic decision trees with a single occurrence per literal is learnable in the uniform model and in polynomial time.*

4 Generalizations

In this final section we argue about general polynomial size decision trees, as well as convex convex combinations. Proofs here are either omitted or simply sketched. However all details that have been left for the complete paper are straightforward to reconstruct.

A *probabilistic decision tree of polynomial size* can be described as follows: Let $q(n)$ be a fixed polynomial. Let T be a binary tree with at most $q(n)$ interior nodes. Consider a labeling of the nodes of T with the variables $x_1, \ldots, x_n$. Consider a labeling of the leaves of T with numbers in [0,1]. Now, as usual, realize that each element $\vec{x}$ of the n-cube follows a path of the tree from the root to a unique leaf. The value of the probabilistic decision tree on $\vec{x}$ is the label of this unique leaf.

As mentioned in Section 2, and exactly as in [LMN89], polynomial-size decision trees are "very far" from the parity function, which results in bounded spectrum and learnability.

Theorem 4.1 [Straightforward Extension of [LMN89]] *The class of probabilistic decision trees of polynomial size has bounded spectrum. Therefore, the class of probabilistic decision trees of polynomial size is learnable in the uniform model and in slightly super-polynomial time.*

PROOF (outline). The proof of Theorem 4.1 follows identically along the lines of proof of the Main Lemma (Lemma 9) in [LMN89] (with the additional ease that the use of Hastad's Switching Lemma is unecessary). In particular:

• First of all, notice that if a probabilistic polynomial size decision tree is hit with a suitable "random restriction", then with high probability the restricted concept is a decision tree of small depth. The reason is, roughly, that every long branch of the tree will be forced and hence "chopped-off" in small depth.

• Second of all, notice that like small depth boolean decision trees in [LMN89], small depth probabilistic decision trees have absolutely bounded spectrum. More precisely, it is easy to see that if a probabilistic decision tree has depth k then $a(S) = 0$ for all S : $|S| > k$.

• Now the next steps are fairly technical, but identical to the manipulations in Lemmas 5 through 9 in [LMN89]. Very roughly, the idea that the random-restriction method suggests is that any poly-size probabilistic decision tree can be written as an average of many other probabilistic decision trees most of which are of small depth. This, in turn, translates to bounded spectrum.

• Once the boundedness of the spectrum has been established, Theorem 4.1 follows by Theorem 2.2.

□

We finally argue about arbitrary convex combinations of bounded spectrum concepts. The weighted arithmetization of k-DNF follows as a special case.

Recall that for functions $g_1, g_2, \ldots, g_N$, a *convex combination* of the g_i's is a sum of the form $\sum_i \lambda_i g_i$ where $\sum_i \lambda_i = 1$ and all λ_i's are in [0,1].

Theorem 4.2 *If $g_1, g_2, \ldots, g_N$ are bounded spectrum probabilistic concepts over n variables (N is arbitrary), and if g is a convex combination of the g_i's, then g is a bounded spectrum probabilistic concept.*

PROOF. First notice that g is indeed a probabilistic concept in the sence that the range of g is indeed [0,1] (g is the average of quantities in [0,1]).

Then realize that for all S's $a_g(S) = \sum_i \lambda_i a_{g_i}(S)$. Hence:

$$\begin{aligned} \sum_{S:|S|>k} a_g^2(S) &= \sum_{S:|S|>k} \left(\sum_i \lambda_i a_{g_i}(S)\right)^2 \\ &\leq \sum_{S:|S|>k} \sum_i \lambda_i a_{g_i}^2(S) \\ &= \sum_i \lambda_i \sum_{S:|S|>k} a_{g_i}^2(S) \\ &\leq \sum_i \lambda_i \text{poly}(n) 2^{-k^c} \\ &\leq \text{poly}(n) 2^{-k^c} \end{aligned} \tag{7}$$

□

Now notice that if p is some *weighted arithmetization of k-DNF*, that is, p is of the form $p(\vec{x}) = \sum_i \lambda_i c_i(\vec{x})$, where the c_i's are products of k variables or their negations, and since $a_{c_i}(S) = 0$ for all c_i's and $|S| > k$, then (7) suggests that $\sum_{S:|S|>k} a_g^2(S) = 0$. For constant k, this suggests that there are only $O(n^k)$ coefficients to be approximated. Therefore:

Theorem 4.3 *The weighted arithmetization of k-DNF is learnable in the uniform model and in polynomial time.*

Furthermore and very interestingly, the strong condition $\sum_{S:|S|>k} a_g^2(S) = 0$ coupled with Mansour's techniques [M90] suggest:

Theorem 4.4 *The weighted arithmetization of k-DNF is distribution-free learnable in polynomial time.*

In this sense, it might be interesting to formalize some natural arithmetization of AC^0 and check if the results in [LMN89] extend.

5 Summary and Open Problems

Here we used Fourier analysis to obtain polynomial-time uniform-learning algorithms for probabilistic decision lists and single literal decision trees. We further observed that straightforward extension of [LMN89] suggests slightly super-polynomial uniform-learning algorithms for arbitrary probabilistic decision trees of polynomial size.

Of course the most challenging question is to extend these results in the distribution-free model, or obtain negative evidence for such extensions (in some sense like [KV89]).

It might also turn out interesting to pursue a careful study of the wide consequences and uses of Nyquist's theorem, and see how much carries over to the finite case.

References

[BEHW86] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth, "Learnability and the Vaprnik-Chervonenkis Dimension", *Journal of the ACM*, 36(4), 1989, pp, 929-965.

[H86] J. Hastad, "Computational Limitations of Small Depth Circuits", Ph.D. Thesis, *MIT Press*, 1986.

[KS90] M. Kearns and R. E. Shapire, "Efficient Distribution-Free Learning of Probabilistic Concepts, " *Proc. of the 31st IEEE Symposium on Foundations of Computer Science*, 1990, pp 382-391.

[KV89] M. Kearns and L. G. Valiant, "Cryptographic Limitations on Learning Boolean Formulae and Finite Automata, " *Proc. of the 21st ACM Symposium of Theory of Computing*, 1989, pp 433-444.

[LMN89] N. Linial, Y. Mansour, and N. Nissan, "Constant Depth Circuits, Fourier Transforms, and Learnability", *Proc. of the 30th IEEE Symposium on Foundations of Computer Science*, 1989, pp 574-579.

[M90] Y. Mansour, "Learning via Fourier Transforms", Spring 1989, preprint.

[R87] R. Rivest, "Learning Decision Lists", *Machine Learning*, 2(3), 1987, pp 229-246.

[V84] L. G. Valiant, "A theory of the Learnable", Communications of the ACM, 27(11), November 1984, pp 1134-1142.

[Y85] A. C. Yao, "Separating the polynomial-Time Hierarchy by Oracles", *Proc. of the 26th IEEE Symposium on Foundations of Computer Science*, 1985, pp 1-10.