Appears in Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory (COLT), 1992.

A Technique for Upper Bounding the Spectral Norm with Applications to Learning

Mihir Bellare[*]

December 26, 1992

Abstract

We present a general technique to upper bound the spectral norm of an arbitrary function. At the heart of our technique is a theorem which shows how to obtain an upper bound on the spectral norm of a decision tree given the spectral norms of the functions in the nodes of this tree. The theorem applies to trees whose nodes may compute *any* boolean functions.

Applications are to the design of efficient learning algorithms and the construction of small depth threshold circuits (or neural nets). In particular, we present polynomial time algorithms for learning $O(\log n)$ term DNF formulas and various classes of decision trees, all under the uniform distribution with membership queries.

[*]Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: `mihir@cs.ucsd.edu`

1 Introduction

Associated to any real-valued function is a number called its spectral norm. Let us begin by saying what it is.

1.1 Definitions

For each $z \in \{0,1\}^n$ we define $\chi_z : \{-1,+1\}^n \to \{-1,+1\}$ by

$$\chi_z(x) \;=\; (-1)^{\frac{1}{4}(1-z_1)(1-x_1)+\cdots+\frac{1}{4}(1-z_n)(1-x_n)} \; .$$

These are the *parity* functions, and they form a basis for the vector space of boolean functions on the n-cube. In other words, every boolean function $f : \{-1,+1\}^n \to \{-1,+1\}$ can be uniquely expressed as a linear combination of the parity functions. This linear combination is the Fourier series of f. We will denote by $\widehat{f}(z)$ the coefficient of χ_z in the Fourier series of f, so that this Fourier series is $f(x) = \sum_{z\in\{-1,+1\}^n} \widehat{f}(z)\chi_z(x)$. One can show that $\widehat{f}(z) = 2^{-n}\sum_{x\in\{-1,+1\}^n} f(x)\chi_z(x)$. The spectral norm of f, which we denote by $L(f)$, is the sum of the absolute values of the coefficients in the Fourier series. That is, $L(f) \stackrel{\text{def}}{=} \sum_{z\in\{-1,+1\}^n} |\widehat{f}(z)|$.

1.2 The Problem

Spectral techniques have many applications in which the spectral norm plays a key role. One example is learning: we know that if $L(f)$ is polynomially bounded then f is learnable (under the uniform distribution with membership queries) in polynomial time [KM]. Another example is in the domain of threshold circuits (or neural nets): we know that if $L(f)$ is polynomially bounded then f can by computed by a threshold circuit of small depth [BS].

To use such theorems, we need to first show that $L(f)$ is small. So it is important to find ways of computing good upper bounds on the spectral norm, and thereby, in particular, identifying the functions which have polynomially bounded spectral norm.

However, bounding the spectral norm seems to be a hard problem. This is perhaps not surprising since $L(f)$ is, after all, a complex object: it is an exponential sized sum, each term of which is itself an exponential sized sum, and estimating all the relevant quantities often defies analysis, even for quite simple functions. But the consequence is that good upper bounds on $L(f)$ are only known either for very simple functions (like parity, AND and OR) or for functions where one can compute $L(f)$ exactly by exploiting a convenient inductive structure (like the comparison function [SB]). And so far there have been no general techniques to compute upper bounds: the approach used is to work directly from the definition.

This paper presents a general technique to upper bound the spectral norm. Below we describe the technique and then some applications.

1.3 Our Solution

The key to our solution is to find a "language" for expressing one function f in terms of other functions $g_1, \ldots, g_m$ which has the property that there is an automatic way to derive a bound on

$L(f)$ from bounds on $L(g_1), \ldots, L(g_m)$. Then by choosing $g_1, \ldots, g_m$ to be functions for which we know bounds on the spectral norm — these can be computed recursively if necessary — we can get a bound on $L(f)$. The difficulty of course is in finding a language which is sufficiently expressive and also has the property that we can prove the necessary theorems which enable us to automatically combine bounds on $L(g_1), \ldots, L(g_m)$ into a bound on $L(f)$. As we will prove below, the language of *universal decision trees* meets both criteria. Let us now make all this more precise and proceed to describe the main theorem.

A universal decision tree over n variables is a (full) binary tree in which each node is labeled by a boolean function $\{-1,+1\}^n \to \{-1,+1\}$. The function from $\{-1,+1\}^n$ to $\{-1,+1\}$ which a tree computes is defined as follows. On any input $x \in \{-1,+1\}^n$, we start at the root and execute the following procedure until we get an output. At an internal node labeled by $g : \{-1,+1\}^n \to \{-1,+1\}$ we branch to the left if $g(x) = -1$ and to the right if $g(x) = +1$. At a leaf labeled by $g : \{-1,+1\}^n \to \{-1,+1\}$ we output $g(x)$ as the value computed by the tree at this input x. We identify the decision tree with the function it computes and the nodes with the functions labeling them, so that we can speak of the spectral norm $L(T)$ of a decision tree T or the spectral norm of a node.

It is crucial in this definition that we allow the nodes to compute *any* boolean function. This means our "language" has maximal expressive power and the main theorem (which we now state) is of maximum generality.

Theorem 1.1 (Main Theorem) *Let T be a universal decision tree. Then*

$$L(T) \;\leq\; \textstyle\sum_{leaves\ l} L(g_l) \prod_{ancestors\ i\ of\ l}\ \frac{1}{2}[L(g_i)+1]$$

where g_i is the function labeling node i of T.

We will call $\sum_{leaves\ l} L(g_l) \prod_{ancestors\ i\ of\ l} \frac{1}{2}[L(g_i)+1]$ the *weight* of T. We note that this weight can be (easily) computed given the spectral norms of the functions in the nodes of T.

This main theorem is the "automatic" procedure we need. It shows how to combine bounds on the norms of the nodes into a bound on the norm of the tree.

The full technique should now be clear. If we want a bound on $L(f)$, we express f as a decision tree whose nodes compute functions whose norms we already have bounds on (and if bounds on the norms of the nodes are not known, they can be obtained recursively). We then compute the weight of the tree. The main theorem says this is an upper bound on $L(f)$. Part of the success of the technique depends, of course, on making the "right" choice of how to express f as a universal decision tree, amongst the many possible ways the given function f can be expressed as a universal decision tree. The examples in later sections will illustrate.

How good is the bound given by our main theorem? Some indication is provided by the fact that for restricted classes of decision trees where bounds on $L(T)$ were known, the bounds our theorem yields are the same or better. In particular for parity decision trees (each internal node is labeled by a parity function χ_s and each of leaf is labeled $+1$ or -1) our theorem yields $L(T) \leq \lceil |T|/2 \rceil$ which is (a slight improvement of) what [KM] had already proved for this special case ($|T|$ is the number of nodes in T). We note however that parity decision trees are not themselves expressive enough to serve as the basis for a general technique to bound spectral norms.

1.4 Applications

Our main theorem has several applications to the design of efficient learning algorithms.

Learning DNF. We address the problem of learning DNF formulas [Va]. Linial, Mansour and Nisan [LMN], Verbeurgt [Ve], and Furst, Jackson and Smith [FJS] consider (ϵ, δ, D) learning of DNF formulas: after seeing a collection of examples of the target f drawn under distribution D, the algorithm must output a hypothesis h which with probability at least $1-\delta$ satisfies $\Pr_{x \in_R D}[f(x) \neq h(x)] \leq \epsilon$. The first two works present quasi-polynomial (in $n, \epsilon^{-1}, \lg \delta^{-1}$) time algorithms for this task when D is the uniform distribution, and the last generalizes to mutually-independent distributions. To date, no polynomial time algorithm for learning (general) DNF is known.

An often studied special case is k-DNF, where the number of literals per term is at most k. Here we will consider a different type of restriction. We allow any number of literals per term, but ask that the number of terms be small.

The model we consider in this paper is that of learning under the uniform distribution given the ability to make membership queries. More precisely, a learning algorithm receives as input $1^n, \epsilon > 0$ and $\delta > 0$, and is allowed to query the target function $f\colon \{-1,+1\}^n \to \{-1,+1\}$. It is required to output a hypothesis h which with probability at least $1-\delta$ satisfies $\Pr_{x \in_R \{-1,+1\}^n}[f(x) \neq h(x)] \leq \epsilon$. The running time of the algorithm is measured in terms of n, ϵ^{-1} and $\lg \delta^{-1}$ (in particular, polynomial time means polynomial in these quantities). For a more formal description of the model we refer the reader to Section 6.1.

Theorem 1.2 *The class of DNF formulas which have $O(\log n)$ terms is learnable in polynomial time.*

More generally, we show that a k-term DNF formula is learnable in time polynomial in $n, 2^k, \epsilon^{-1}$, and $\lg \delta^{-1}$.

Independently of this work, Blum and Rudich [BR] present an algorithm which uses membership and equivalence queries to (exactly) learn a k-clause DNF formula in time polynomial in n and 2^k.

Learning Decision Trees. We present a general result about learning decision trees.

Theorem 1.3 *If a class of (universal) decision trees has polynomially bounded weight then it is learnable in polynomial time.*

Several special cases of this theorem are of interest. See Section 6.1 for details on these applications.

All of the above learning results are obtained by using our main theorem to bound the spectral norm of the class of functions in question and then applying the learning algorithm of [KM].

Minimizing the depth of neural nets. Threshold circuits (or neural nets) are circuits whose gates are linear threshold functions (see Section 6.4 for full definitions). The depth of the circuit is an important measure of efficiency, and much work in the area has concentrated on depth minimization [HMPST, Br, BS, SB]. Spectral theory has provided an important tool for this task with a theorem which says that functions of polynomially bounded spectral norm have depth two threshold circuits [BS]. Combining this with our main theorem yields a novel tool.

Theorem 1.4 *If a boolean function is expressible as a universal decision tree of polynomially bounded weight then it is computable by a depth two threshold circuit.*

Moreover, in the design of a neural net it is often the case that one needs to combine simple functions into a more complex circuit. The above theorem is particularly suited to this type of application, since it automatically provides a way to get small depth circuits for combinations of simple functions.

We note that the theorem of [BS], and in consequence ours, are non-constructive.

The results of this paper appeared previously in [Be1] and [Be2].

2 Preliminaries

"Boolean" for us means ± 1 valued. A function $f\colon \{-1,+1\}^n \to \mathsf{R}$ is boolean if its range is $\{-1,+1\}$. The sign of $x \in \mathsf{R}$, denoted *sign*(x), is -1 if $x < 0$, undefined if $x = 0$, and $+1$ if $x > 0$. The sign of $f\{-1,+1\}^n \to \mathsf{R}$, denoted *sign*$(f)$ is the map defined by $x \mapsto \mathit{sign}(f(x))$.

We identify a string $z \in \{-1,+1\}^n$ with the set $\{\, i : z_i = -1 \,\}$. With this convention, we apply set-theoretic notation to strings, using expressions like "$i \in z$" or "$y \triangle z$" ($\triangle$ is the symmetric difference) where $y, z \in \{-1,+1\}^n$ and $1 \le i \le n$.

We define the inner product of $f, g\colon \{-1,+1\}^n \to \mathsf{R}$ by

$$\langle f, g\rangle \;=\; 2^{-n} \textstyle\sum_{x \in \{-1,+1\}^n} f(x)g(x) \;.$$

Note that $\langle f, g\rangle = \mathbf{E}[fg]$ where the expectation is over the uniform distribution on the inputs. The norm associated to this inner product is

$$\|f\| \;=\; \sqrt{\langle f, f\rangle} \;.$$

The *parity functions* are the functions $\chi_z : \{-1,+1\}^n \to \{-1,+1\}$ defined for $z \in \{-1,+1\}^n$ by $\chi_z(x) = (-1)^{\frac{1}{4}(1-z_1)(1-x_1)+\cdots+\frac{1}{4}(1-z_n)(1-x_n)}$. That is, $\chi_z(x)$ is -1 if the number of indices i at which $z_i = x_i = -1$ is odd, and 1 otherwise. It is well known that $\{\chi_z\}_{z \in \{-1,+1\}^n}$ is an orthonormal basis for the vector space of real valued functions on $\{-1,+1\}^n$ (the orthonormality is with respect to the inner product defined above). A useful property of the parity functions is that $\chi_a\chi_b = \chi_{a \triangle b}$.

Now suppose $f\colon \{-1,+1\}^n \to \mathsf{R}$. Then

$$f \;=\; \textstyle\sum_{z \in \{-1,+1\}^n} \widehat{f}(z)\chi_z$$

where $\widehat{f}(z) \;=\; \langle f, \chi_z\rangle \;=\; 2^{-n} \sum_{x \in \{-1,+1\}^n} f(x)\chi_z(x)$. This (unique) expansion of f in terms of the parity basis is its *Fourier series*. The Fourier coefficients of f are the real numbers $\widehat{f}(z)$

($z \in \{-1,+1\}^n$). The sequence of Fourier coefficients is also called the *spectrum* of f.[†] The *Fourier transform* is the operator $\mathcal{F}$ defined by $\mathcal{F}(f) = \widehat{f}$. This operator is linear: $\mathcal{F}(af+bg) = a\mathcal{F}(f) + b\mathcal{F}(g)$ for real a, b and functions f, g: $\{-1,+1\}^n \to \mathsf{R}$. The *spectral norm* (or just *norm*) of f is

$$L(f) \overset{\text{def}}{=} \textstyle\sum_{z\in\{-1,+1\}^n} |\widehat{f}(z)| \ .$$

In the context of norms, "small" means polynomially bounded (as a function of n) and "large" means not small. Parseval's identity

$$\textstyle\sum_{z\in\{-1,+1\}^n} \widehat{f}(z)^2 \ = \ \|f\|^2$$

follows directly from the orthonormality of the basis. It follows that if f is boolean then $\sum_{z\in\{-1,+1\}^n} \widehat{f}(z)^2 = 1$. A consequence of this last fact is the following

Proposition 2.1 *$L(f) \le 2^{n/2}$ for any boolean function $f : \{-1,+1\}^n \to \{-1,+1\}$.*

3 Main Theorem

Here we present our general technique for upper bounding the spectral norm of a universal decision tree.

Definition 3.1 *A* universal decision tree *with input length n is a (full) binary tree*[†] *in which each node is labeled by a function from $\{-1,+1\}^n$ to $\{-1,+1\}$.*

Note that we do not restrict the functions labeling the nodes: *any* boolean function is allowed.

T computes (or defines) a function from $\{-1,+1\}^n$ to $\{-1,+1\}$ in a natural way which we now describe. Suppose $x \in \{-1,+1\}^n$. The value of $T(x)$ is obtained by repeating the following procedure, starting at the root, until an output is obtained:

- If the current position is an internal node, then branch to the left if $g(x) = -1$ and to the right if $g(x) = +1$, where g is the function labeling this internal node.
- If the current position is a leaf, then output $g(x)$ where g is the function labeling this leaf.

We identify the decision tree with the function it computes and the nodes with the functions labeling them, so that we can speak of the spectral norm $L(T)$ of a universal decision tree T or the spectral norm of a node. $|T|$ will denote the number of nodes in T.

In order to formulate our main theorem we first need some definitions. Indeed, most of the work here lies in making the correct definitions and once this is done the main theorem will follow quite easily.

We assign to each function a number which we will call its *degree*. The degrees of the constituent nodes will enable us to measure the spectral norm of a decision tree. The definition itself is very simple:

[†] Some intuition about the Fourier coefficients may be gathered by observing that $\widehat{f}(z) = \Pr[f(x) = \chi_z(x)] - \Pr[f(x) \neq \chi_z(x)]$, the probabilities being over a random choice of the input x.

[†] By this we mean that every node has either two children or no children.

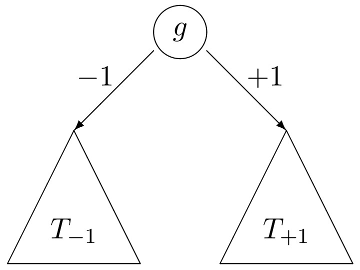

Figure 1: The decision tree T

Definition 3.2 *The* degree *of g: $\{-1,+1\}^n \to \{-1,+1\}$ is $deg(g) \overset{\text{def}}{=} \frac{1}{2}[L(g)+1]$.*

So the degrees are, strictly speaking, the spectral norms themselves: it is just more convenient to "translate" slightly. Note that $L(g) \geq 1$ for any boolean g. So 1 is also the minimal degree of any boolean function.

We will assign to each decision tree a weight which is computed as a function of the spectral norms (or, more precisely, the degrees) of its constituent nodes.

Definition 3.3 *The* weight *$w(T)$ of a universal decision tree T is defined by induction as follows:*

- *If $|T| = 1$ then we let $w(T) = L(g)$, where g is the function labeling the root of T.*
- *If $|T| \geq 3$ then we let $w(T) = deg(g)[w(T_{-1}) + w(T_{+1})]$, where g is the function labeling the root of T, and T_{-1} and T_{+1} are the left and right subtrees of T respectively (cf. Figure 1).*

Unraveling the recursion, we can see that the weight of a tree can be computed as follows. First, for each leaf l, multiply the spectral norm of the function g_l labeling this leaf by the product of the degrees of the functions labeling the ancestors of l (the ancestors of l are the nodes on the path from l to the root, but excluding l). Now sum these quantities over all leaves l. Thereby we obtain the expression quoted in Section 1.3. For future reference, let us state this as a Proposition.

Proposition 3.4 *Let T be a universal decision tree. Then*

$$w(T) \ = \ \sum_{leaves\ l} L(g_l) \prod_{ancestors\ i\ of\ l} deg(g_i) \ ,$$

where g_i is the function labeling node i.

Proof: By induction. ■

We need a couple of (easy) lemmas before we can prove the main theorem. The first says that multiplying a function f by a parity function does not change its spectral norm.

Lemma 3.5 *$L(f\chi_s) = L(f)$ for any function f.*

Proof: Recall that $s \triangle z = (s \cup z) - (s \cap z)$ is the symmetric difference of $s, z \in \{-1,+1\}^n$. We have

$$\begin{aligned} \widehat{f\chi_s}(z) &= 2^{-n} \textstyle\sum_x f(x)\chi_s(x)\chi_z(x) \\ &= 2^{-n} \textstyle\sum_x f(x)\chi_{s \triangle z}(x) \\ &= \widehat{f}(s \triangle z) \ . \end{aligned}$$

The lemma follows from the fact that the map $z \mapsto s \triangle z$ is a permutation of $\{-1,+1\}^n$. ■

The next lemma shows how to bound the spectral norm of sums and products of functions in terms of the spectral norms of these functions.

Lemma 3.6 *Suppose* $f, g\colon \{-1,+1\}^n \to \mathsf{R}$. *Then*

(1) $L(f+g) \leq L(f) + L(g)$

(2) $L(fg) \leq L(f)L(g)$.

Proof: Using the linearity of the Fourier transform and the triangle inequality we have

$$\begin{aligned} L(f+g) &= \textstyle\sum_z |\widehat{f+g}(z)| \\ &= \textstyle\sum_z |\widehat{f}(z) + \widehat{g}(z)| \\ &\leq \textstyle\sum_z \left(|\widehat{f}(z)| + |\widehat{g}(z)|\right) \\ &= L(f) + L(g) \,. \end{aligned}$$

This establishes (1). To prove (2), set $a_z = \widehat{g}(z)$. Then making use of (1) we have

$$\begin{aligned} L(fg) &= L(f \cdot \textstyle\sum_z a_z \chi_z) \\ &= L(\textstyle\sum_z a_z f \chi_z) \\ &\leq \textstyle\sum_z L(a_z f \chi_z) \\ &= \textstyle\sum_z |a_z| L(f\chi_z) \,. \end{aligned}$$

But by Lemma 3.5 this equals

$$\textstyle\sum_z |a_z| L(f) \;=\; L(f) \sum_z |a_z| \;=\; L(f)L(g)$$

as desired. ■

We are now ready to present the main theorem. It is very simply stated: it says that the spectral norm of a decision tree is at most its weight.

Theorem 3.7 (Main Theorem) $L(T) \leq w(T)$ *for any universal decision tree* T.

Proof: The proof is by induction (on the structure of the tree). For the base case of T consisting of a single node labeled by g we have $L(T) = L(g) = w(T)$. Now suppose T has ≥ 3 nodes. Let g be the function labeling the root of T, and let T_{-1} and T_{+1} be its left and right subtrees respectively (cf. Figure 1). Observe that

$$T(x) \;=\; \tfrac{1}{2}[1 - g(x)]T_{-1}(x) + \tfrac{1}{2}[1 + g(x)]T_{+1}(x) \,.$$

That is, $T = \frac{1}{2}T_{-1} + \frac{1}{2}T_{+1} - \frac{1}{2}gT_{-1} + \frac{1}{2}gT_{+1}$. Applying Lemma 3.6 (1) we get that $L(T)$ is at most

$$L(\tfrac{1}{2}T_{-1}) + L(\tfrac{1}{2}T_{+1}) + L(-\tfrac{1}{2}gT_{-1}) + L(\tfrac{1}{2}gT_{+1}) \,.$$

$g : \{-1,+1\}^n \to \{-1,+1\}$	$L(g)$	$deg(g)$
Parity: $\chi_s \ \ (s \in \{-1,+1\}^n)$	1	1
Conjunction of literals: $\bigwedge_{i\in s}\tilde{x}_i \ \ (s \in \{-1,+1\}^n)$	≤ 3	≤ 2
Disjunction of literals: $\bigvee_{i\in s}\tilde{x}_i \ \ (s \in \{-1,+1\}^n)$	≤ 3	≤ 2
Comparison	$O(n)$	$O(n)$
(Each output bit of) Addition	$O(n)$	$O(n)$

Figure 2: Spectral Norms and Degrees of some well known functions

By Lemma 3.6 (2), this is at most

$$\begin{aligned}&\tfrac{1}{2}L(T_{-1}) + \tfrac{1}{2}L(T_{+1}) + \tfrac{1}{2}L(g)L(T_{-1}) + \tfrac{1}{2}L(g)L(T_{+1})\\ &= \tfrac{1}{2}\left[L(g)+1\right]L(T_{-1}) \ + \ \tfrac{1}{2}\left[L(g)+1\right]L(T_{+1})\\ &= deg(g)\left[L(T_{-1}) + L(T_{+1})\right] \ .\end{aligned}$$

But by induction $L(T_{-1}) \leq w(T_{-1})$ and $L(T_{+1}) \leq w(T_{+1})$ so the above is $\leq deg(g)[w(T_{-1})+w(T_{+1})]$ which by definition is $w(T)$. This concludes the induction. ■

4 Degrees of boolean functions

To use the main theorem we need to be able to compute weights. And to compute weights we need to know the degrees of the functions involved. So it is worth developing some understanding of the degrees of boolean functions. Below we study the degrees of some well known boolean functions. We will use the facts developed here in later sections. We begin by observing that the parity functions are exactly those of minimal degree.

Proposition 4.1 *Suppose g is boolean. Then $deg(g) = 1$ if and only if $L(g) = 1$ if and only if $g = \pm\chi_s$ for some s.*

Proof: It is clear that $deg(g) = 1$ if and only if $L(g) = 1$, and it is also clear that $L(g) = 1$ if $g = \pm\chi_s$. To complete the proof it suffices to show that if $L(g) = 1$ then $g = \pm\chi_s$ for some s. So suppose $L(g) = 1$. Then for any x we have

$$\begin{aligned}1 &= |g(x)|\\ &= |\textstyle\sum_z \widehat{g}(z)\chi_z(x)|\\ &\leq \textstyle\sum_z |\widehat{g}(z)\chi_z(x)|\\ &= L(g)\\ &= 1\ ,\end{aligned}$$

and thus $|\sum_z \widehat{g}(z)\chi_z(x)| = \sum_z |\widehat{g}(z)\chi_z(x)|$. It follows that for each x either $\widehat{g}(z)\chi_z(x) \geq 0$ for all z or $\widehat{g}(z)\chi_z(x) \leq 0$ for all z. Moreover the former happens when $g(x) = 1$ and the latter when $g(x) = -1$, because $g(x) = \sum_z \widehat{g}(z)\chi_z(x)$. So fixing any z such that $\widehat{g}(z) \neq 0$ (at least one such must exist) we can conclude that $sign(\widehat{g}(z)) = \chi_z(x)$ if $g(x) = 1$ and $sign(\widehat{g}(z)) = -\chi_z(x)$ if $g(x) = -1$. But this is true for all x, and thus g is either χ_z or $-\chi_z$. ■

Next we consider conjunctions $\bigwedge_{i\in s}\tilde{x}_i$ and disjunctions $\bigvee_{i\in s}\tilde{x}_i$ of literals over arbitrary subsets $s \in \{-1,+1\}^n$ of the variables (recall that we identify $s \in \{-1,+1\}^n$ with $\{\, i : s_i = -1 \,\}$). The literal $\tilde{x}$ is either x or $\bar{x}$. As usual we are taking the outputs to be ± 1 (-1 is true and 1 is false). One can show that $L(\bigwedge_{i\in s}\tilde{x}_i), L(\bigvee_{i\in s}\tilde{x}_i) \leq 3$. So $deg(\bigwedge_{i\in s}\tilde{x}_i), deg(\bigvee_{i\in s}\tilde{x}_i) \leq 2$ (note that the bound is independent of the number of literals in the clause). So these functions are just a step higher than parity in the degree hierarchy.

Let us now consider arithmetic functions. The comparison function $C_n : \{-1,+1\}^n \to \{-1,+1\}$ is defined for even n by $C_n(xy) = 1$ if $x \geq y$ and -1 otherwise, where $x, y \in \{-1,+1\}^{n/2}$ are identified with the integers $\frac{1}{2}\sum_{i=1}^{n/2}(1-x_i)2^{\frac{n}{2}-i}$ and $\frac{1}{2}\sum_{i=1}^{n/2}(1-y_i)2^{\frac{n}{2}-i}$ respectively. It is well known (cf. [SB]) that $L(C_n) = \frac{n}{2} + 1$. Similarly, each output bit of the addition function can be shown to have degree $O(n)$.

These facts are summarized in Figure 2.

5 Special classes of decision trees

Most important applications of the main theorem are derived by looking at some subset of the class of universal decision trees. It is convenient to have the following terminology.

Definition 5.1 *Let B be a set of boolean functions. A decision tree over basis B is a universal decision tree all of whose nodes compute functions in B.*

For example, *parity* decision trees are decision trees over the basis consisting of $\{\pm\chi_s\}_{s\in\{-1,+1\}^n}$ whose leaves are labeled by constants (1 and -1). More generally, we may consider decision trees over the basis of parity, $\bigwedge_{i\in s}\tilde{x}_i$, and $\bigvee_{i\in s}\tilde{x}_i$ functions. We note that $w(T) = \lceil |T|/2 \rceil$ for any parity decision tree T (cf. Proposition 3.4 and Proposition 4.1). Thus the main theorem implies that $L(T) \leq \lceil |T|/2 \rceil$ for any parity decision tree T (cf. [KM]).

6 Applications

One of the main tasks in computational learning theory is to identify which functions are efficiently learnable. Our main theorem provides a tool to enlarge the class of functions known to be efficiently learnable.

6.1 Preliminaries

The model we consider is that of learning boolean functions with high probability given the ability to query the target. More precisely, the learning algorithm receives as input $1^n, \epsilon, \delta$ and, as an oracle, the function $f : \{-1,+1\}^n \to \{-1,+1\}$ which it is trying to learn. Its output is (an encoding of) a *hypothesis* $h : \{-1,+1\}^n \to \{-1,+1\}$. The running time of the learning algorithm is measured in terms of n, ϵ^{-1} and $\log \delta^{-1}$ (in particular, polynomial time means polynomial in these quantities). We say that $\mathcal{A}$ learns a class $\mathcal{C}$ of boolean functions if the output h of $\mathcal{A}$ on input $1^n, \epsilon, \delta$ satisfies $Err_h(f) \leq \epsilon$ with probability $\geq 1-\delta$, for every $f : \{-1,+1\}^n \to \{-1,+1\}$ from $\mathcal{C}$ and every n and $\epsilon, \delta > 0$. Here $Err_h(f) \overset{\text{def}}{=} \Pr[f(x) \neq h(x)]$ (the probability is over a random choice of $x \in \{-1,+1\}^n$) is the *error* of the hypothesis h with respect to the target f and the probability is over the coin tosses of $\mathcal{A}$.

We say that l: $\mathsf{N} \to \mathsf{N}$ is a bound on the spectral norm of a class of functions $\mathcal{C}$ if $L(f) \leq l(n)$ for all $f : \{-1,+1\}^n \to \{-1,+1\}$ in $\mathcal{C}$. The main result we will exploit is the following.

Theorem 6.1 [KM] *A class of boolean functions is learnable in time polynomial in $l(n), n, \epsilon^{-1}$ and $\lg \delta^{-1}$, where l is a bound on the spectral norm of the class.*

6.2 Learning DNF

A DNF formula is a disjunction $C_1 \vee \ldots \vee C_k$ of terms $C_1, \ldots C_k$ where each term is a conjunct of literals: $C_j = \bigwedge_{i \in s_j} \tilde{x}_i$ for some $s_j \in \{-1,+1\}^n$.

Theorem 6.2 *The class of $O(\log n)$-term DNF formulas is learnable in polynomial time.*

Proof: It suffices to show that the class of k-term DNF formulas has spectral norm at most $2^{O(k)}$. To do this, we "simulate" the DNF formula $C_1 \vee \ldots \vee C_k$ by a decision tree over the basis $\{ \bigwedge_{i \in s} \tilde{x}_i : s \in \{-1,+1\}^n \}$, and then bound the spectral norm of the decision tree.

The decision tree is defined (inductively) as follows. If $k = 1$ it consists of a single node computing C_1. Otherwise, the root computes C_1 and its left son is a leaf labeled "-1" while its right son is the root of a tree computing $C_2 \vee \ldots \vee C_k$. (In other words, our tree is just the "decision list" corresponding to the formula). We recall (cf. Figure 2) that the spectral norm of a $\bigwedge_{i \in s} \tilde{x}_i$ is at most 3 and the degree of a $\bigwedge_{i \in s} \tilde{x}_i$ is at most 2. By Proposition 3.4, the weight of the tree is then at most $3 \cdot 2^{k-1} + \sum_{i=1}^{k-1} 2^i = 2^{O(k)}$. So by the main theorem, the spectral norm of this tree is at most $2^{O(k)}$. That is, $L(C_1 \vee \ldots \vee C_k) \leq 2^{O(k)}$. ■

More generally, we have shown that the class of k-clause DNF formulas is learnable in time polynomial in $n, 2^k, \epsilon^{-1}$, and $\lg \delta^{-1}$.

We note that the $2^{O(k)}$ bound on the spectral norm of a k-clause DNF formula that we got by applying our main theorem is essentially optimal. To see this, observe that the DNF formula $(x_1 \wedge x_2) \vee \ldots \vee (x_{2k-1} \wedge x_{2k})$ has spectral norm $\geq 2^{\Omega(k)}$. In particular this means that for DNF with a super-logarithmic number of clauses one may not be able to prove polynomial time learnability with these same techniques.

6.3 Learning Decision Trees

The original motivation of the learning algorithm of [KM] was to learn parity decision trees with polynomially many nodes. We can extend this to learn many larger classes of decision trees.

Theorem 6.3 *Let $\mathcal{C}$ be a class of universal decision trees with polynomially bounded weight. Then $\mathcal{C}$ is learnable in polynomial time.*

Proof: Follows directly from our Main Theorem and Theorem 6.1. ■

For example, let $\mathcal{C}$ be a class of decision trees over the basis $\{ \chi_s, \bigwedge_{i\in s}\tilde{x}_i, \bigvee_{i\in s}\tilde{x}_i : s \in \{-1,+1\}^n \}$ with the property that the number of $\bigwedge_{i\in s}\tilde{x}_i$ or $\bigvee_{i\in s}\tilde{x}_i$ nodes on any root to leaf path is at most $O(\log n)$. Then $\mathcal{C}$ is learnable in polynomial time. This follows from Theorem 6.3 once we note that (by Proposition 3.4 and the fact that $deg(\chi_s) = 1$ and $deg(\bigwedge_{i\in s}\tilde{x}_i), deg(\bigvee_{i\in s}\tilde{x}_i) \leq 2$) the weight is polynomially bounded in this case. Similarly we can allow comparison nodes (a constant number per root to leaf path) or other kinds of nodes in appropriate proportion to their degree, and still maintain polynomial time learnability.

6.4 Neural Nets

We say that f is a linear threshold function if there exist polynomially bounded integer weights $w_0, w_1, \ldots, w_n$ such that

$$f(x) \;=\; sign\;(w_0 + \textstyle\sum_{i=1}^{n} w_i x_i)$$

A threshold circuit (or neural net) is a circuit in which every gate computes a linear threshold function. The class of boolean function computable by depth d, polynomial sized threshold circuits is usually denoted $\widehat{\mathrm{LT}}_d$.

Spectral theory has provided an important tool for proving the existence of small depth threshold circuits. Bruck and Smolensky [BS] have shown that if a boolean function has polynomially bounded spectral norm then it is computable by a depth two threshold circuit. This theorem (and extensions of it) are exploited in [BS, SB] to prove that many functions (comparison and addition are examples) are in $\widehat{\mathrm{LT}}_2$. By combining the result of [BS] with our main theorem we get a new method of proving the existence of small depth threshold circuits:

Theorem 6.4 *If a boolean function is expressible as a universal decision tree of polynomially bounded weight then it is computable by a depth two threshold circuit.*

In the design of a neural net it is often the case that one needs to combine simple functions into a more complex circuit. Our main theorem is particularly suited to this type of application, since it automatically provides a way to get small depth circuits for combinations of simple functions.

7 Extensions

Furst, Jackson and Smith [FJS] introduce the notion of *mutually independent* distributions and show how to extend the spectral techniques of Linial, Mansour and Nisan [LMN] (used for learning DNF under the uniform distribution) to the case of mutually independent distributions. Our techniques and results can be similarly extended to the case of mutually independent distributions. We discuss these extensions briefly here. For more details the reader is referred to [Be1].

A probability distribution $q : \{-1,+1\}^n \to [0,1]$ is *mutually independent* if the random variables $x_1, \ldots, x_n$ are independent. Given a mutually independent distribution q one can define, with respect to q, an inner product, an orthonormal basis, and finally a Fourier series for every function. We will denote by $L_q(f)$ the spectral norm of f under the Fourier series given by mutually independent distribution q. We call it the q-spectral norm. The exact definitions of these quantities are not necessary here; for these the reader is referred to [FJS].

To state the extension of our main theorem to the q-framework we need to redefine the degree. The rest will follow.

Definition 7.1 *Let q be a mutually independent distribution. The q-degree of g: $\{-1,+1\}^n \to \{-1,+1\}$ is defined to be the least positive real number d with the property that $L_q(fg) \le (2d-1)L_q(f)$ for all f: $\{-1,+1\}^n \to \{-1,+1\}$. We denote the q-degree of g by $deg_q(g)$.*

Note that when q is the uniform distribution this definition of the degree coincides with the one in Section 3. To see this, let d be the least positive real number with the property that $L(fg) \le (2d-1)L(f)$ for all $f : \{-1,+1\}^n \to \{-1,+1\}$. Setting f to be a parity function and using Lemma 3.5 we get $2d-1 \ge L(g)$. But for any f: $\{-1,+1\}^n \to \{-1,+1\}$, Lemma 3.6 (2) says that $L(fg) \le L(f)L(g)$ and so $2d-1 \le L(g)$. Together these imply $d = \frac{1}{2}[L(g)+1] = deg(g)$, as claimed. However we do not know whether or not $deg_q(g) = \frac{1}{2}[L_q(g)+1]$ in general.

The q-weight of a decision tree can be defined just like in Definition 3.3, except that we would use the q-spectral norm and q-degree in place of the spectral norm and degree respectively. We denote the q-weight of a decision tree T by $w_q(T)$. The main theorem then generalizes in the natural way.

Theorem 7.2 *Let q be a mutually independent distribution. Then $L_q(T) \le w_q(T)$ for any universal decision tree T.*

We omit the proof which is identical to that of Theorem 3.7.

It was observed in [Be1] that the learning algorithm of [KM] also extends to the case of mutually independent distributions. Thus in combination with Theorem 7.2 we have another tool for learning under mutually independent distributions.

Acknowledgements

I thank Marek Karpinski and Ron Rivest for helpful discussions.

Work done while the author was at MIT.

References

[Be1] M. Bellare. The Spectral Norm of Finite Functions. *MIT Laboratory for Computer Science Technical Report MIT/LCS/TR-495*, February 1991.

[Be2] M. Bellare. A Technique for Upper Bounding the Spectral Norm with Applications to Learning. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ACM, 1992.

[BR] A. Blum and S. Rudich. Fast Learning of k-term DNF formulas with queries. *Proceedings of the 24th Annual Symposium on the Theory of Computing*, ACM, 1992.

[Br] J. Bruck. Harmonic Analysis of Polynomial Threshold Functions. *SIAM J. Discrete Math.* **3**(2), 168-177 (1990).

[BS] J. Bruck and R. Smolensky. Polynomial Threshold Functions, AC^0 Functions, and Spectral Norms. *Proceedings of the 31st Symposium on Foundations of Computer Science*, IEEE, 1990.

[FJS] M. Furst, J. Jackson and S. Smith. Learning AC^0 Functions Sampled under Mutually Independent Distributions. Manuscript (October 1990).

[HMPST] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy and G. Turan. Threshold Circuits of Bounded Depth. *Proceedings of the 28th Symposium on Foundations of Computer Science*, IEEE, 1987.

[KKL] J. Kahn, G. Kalai and N. Linial. The Influence of Variables on Boolean Functions. *Proceedings of the 29th Symposium on Foundations of Computer Science*, IEEE, 1988.

[KM] E. Kushilevitz and Y. Mansour. Learning Decision Trees using the Fourier Spectrum. *Proceedings of the 23rd Annual Symposium on the Theory of Computing*, ACM, 1991.

[LMN] N. Linial, Y. Mansour and N. Nisan. Constant Depth Circuits, Fourier Transform, and Learnability. *Proceedings of the 30th Symposium on Foundations of Computer Science*, IEEE, 1989.

[SB] K. Siu and J. Bruck. On the Power of Threshold Circuits with Small Weights. Manuscript.

[Va] L. Valiant. A Theory of the Learnable. *Communications of the ACM* **27**(11), 1134-1142 (1984).

[Ve] K. Verbeurgt. Learning DNF under the uniform distribution in polynomial time. *Proceedings of the Second Annual Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers Inc. (1989).