1 Last time and today

Last Time:

- Completed the proof that any $s$-term Disjunctive Normal Form logic function (DNF) has a Polynomial Threshold Function (PTF) of degree $O(n^{1/3} \cdot \log s)$
- Introduced the notion of intersections of halfspaces
- Proved that if $h_1$ and $h_2$ are weight-$w$ Linear Threshold Functions (LTFs) then $h_1 \land h_2$ has a PTF degree at most $O(\log w)$

Today’s Agenda:

- Prove two lemmas from last lecture which were used to prove the LTF result stated above.
- Complete first unit on PAC learning via PTFs of bounded degree
- Begin the second unit on uniform distribution PAC
- Introduce the method of Fourier analysis for Boolean functions

Relevant Readings:

- M. Minsky, S. Papert (1968). Perceptrons
2 COMPLETION OF UNIT 1: LOW DEGREE PTF METHODS

2 Completion of Unit 1: Low Degree PTF Methods

We begin by proving two lemmas which were stated without proof in Lecture 3.

2.1 Proof of Lemmas from Lecture 3

Recall that, in the course of proving that the intersection of weight-\(w\) halfspaces has a degree \(O(\log w)\) PTF, we approximated \(\text{sign}(x)\) using the function \(S_t(x)\) on the domain \(1 \leq |x| \leq W = 2^t\), which was constructed as a rational function of polynomials of the form:

\[
P_t(x) = (x - 1)(x - 2)^2(x - 4)^2 \cdots (x - 2^t)^2.
\]

We now prove the two Lemmas which were used to derive the \(O(\log w)\) result from last lecture.

**Lemma 1.** If \(x \in [1, 2^t]\) then \(4P_t(x) \leq -P_t(-x)\).

**Proof.** Factoring \(-1\) terms and simplifying we have:

\[
-P_t(-x) = -(-x - 1)(-x - 2)^2 \cdots (-x - 2^t)^2 = (x + 1)(x + 2)^2 \cdots (x + 2^t)^2,
\]

and since \(x > 0\) it is clear that \(P_t(x) \leq -P_t(-x)\) termwise. Hence it suffices to show that \(4(x - 2^k)^2 \leq (x + 2^k)^2\) for a single term. First note that if \(x = 2^s\) for some \(s \in \{0, \ldots, t\}\) then this relation is trivially verified. Otherwise, we may choose \(k\) such that \(x \in [2^{k-1}, 2^k)\). The result follows since

\[
4(x - 2^k)^2 = 4(2^k - x)^2 \leq 4(2^{k-1})^2 = 2^{2k} < (x + 2^k)^2.
\]

\[\square\]

**Lemma 2.** Define

\[
S_t(x) := \frac{P_t(-x) - P_t(x)}{P_t(-x) + P_t(x)}.
\]

Then

i) \(x \in [1, 2^t] \Rightarrow S_t(x) \in [1, 5/3]\)

ii) \(x \in [-2^t, -1] \Rightarrow S_t(x) \in [-5/3, -1]\)
2 COMPLETION OF UNIT 1: LOW DEGREE PTF METHODS

Proof. Claim i): Let $x \in [1, 2^t]$. If $P_t(x) = 0$ then $S_t(x) = 1$ and the claim is satisfied. Suppose $P_t(x) \neq 0$. Then we may divide by $-P_t(x)$ to write:

$$S_t(x) = \frac{-P_t(-x)/P_t(x) + 1}{-P_t(-x)/P_t(x) - 1}.$$

Since $\frac{x + 1}{x - 1} > 1$ is a monotonic decreasing function on $\mathbb{R}^+$ and $-P_t(-x)/P_t(x) \geq 4$ by Lemma 1, we have

$$1 < S_t(x) = \frac{-P_t(-x)/P_t(x) + 1}{-P_t(-x)/P_t(x) - 1} \leq \frac{4 + 1}{4 - 1} = \frac{5}{3},$$

so that the claim is satisfied for $P_t(x) \neq 0$ as well.

Claim ii): Since $S_t(x)$ is an odd function by construction, $i) \Rightarrow ii)$.

2.2 Consequences and Generalizations

Having proved that the intersection of two weight-$w$ halfspaces has a PTF of degree $O(\log w)$, our results from previous lectures tell us that this intersection can be PAC-learned in time $n^{O(\log w)}$.

One may naturally wonder if these results can be extended and generalized and we state two of these.

Theorem 3. (k halfspaces) Let $h_1, ..., h_k$ be a collection of weight-$w$ halfspaces. Then $h_1 \land ... \land h_k$ has a corresponding PTF of degree $O(k \log k \log w)$.

Theorem 4. (Arbitrary Boolean Functions) Let $g : \{0, 1\}^k \to \{0, 1\}$ be any Boolean function, and suppose $h_1, ..., h_k$ is a collection of weight-$w$ halfspaces. Then $g(h_1, ..., h_k)$ has a corresponding PTF of degree $O(k^2 \log w)$.

2.3 Lower Bound Results

So far we have considered upper-bounds on PTF degree for intersections of weight-$w$ halfspaces. We now turn to the problem of finding lower bounds.
2.3.1 Poly(n) Weight Halfspaces

The question of a lower bound on PTF degree for polynomial weight halfspaces was only resolved recently, more than 40 years after the initial results were published. We summarize the milestone results below:

- A bound of $\omega(1)$ was due to Minsky and Papert (1968)
- O’Donnell and Servedio (2003) proved an $\omega\left(\frac{\log n}{\log \log n}\right)$ lower bound
- Sherstov (2009): $\Omega(\log n)$

In particular, Sherstov’s result showed that a bound of $O(\log n)$ is the best upper bound we can find, and effectively solved the complexity problem for intersections of Poly(n) weight halfspaces.

2.3.2 Arbitrary Halfspaces

Sherstov also showed that for two arbitrary halfspaces (where no restriction on weight is given), there is a lower bound $\Omega(n)$, indicating that the PTF degree methods we have been using so far will not produce a non-trivial running time result for this class. Whether there is an algorithm to learn the intersection of arbitrary halfspaces which leads to non-trivial results remains an open question:

Open Question: Can we learn the intersection of 2 arbitrary halfspaces in time $2^{o(n)}$?

3 Unit 2: Uniform Distribution Learning

3.1 Motivation

We can summarize our main approach to PAC learning so far as follows:

1. Prove that every $c \in C$ has a low degree PTF $p : \{0,1\}^n \rightarrow \mathbb{R}$ such that $\text{sign}(p(x)) = c(x)$ for every $x \in \{0,1\}^n$

2. Use Linear Programming to find or sufficiently approximate $p$, with the speed of the algorithm depending on the degree of the PTF

Note that in step 1, we require $\text{sign}(p(x))$ and $c(x)$ to agree on all of $\{0,1\}^n$. Naturally, we may consider whether loosening this restriction so that $\text{sign}(p(x)) = c(x)$ for most $x \in \{0,1\}^n$ might lead to lower degree $p$, and hence a faster algorithm.
However, the level generality invoked by letting $D$ be arbitrary makes it difficult to derive useful results when $\text{sign}(p(x)) \neq c(x)$ for some $x \in \{0, 1\}^n$. This is due to an abundance of pathological distributions. As a simple example, suppose $x_i \in \{0, 1\}^n$ such that $\text{sign}(p(x_i)) \neq c(x_i)$. Then $err^\delta(h, c) = 1$ for the measure $\delta_i$ which places unit mass on $x_i$. So we would like to restrict our distributions in some way to avoid this issue.

### 3.2 Uniform Distribution PAC Learning

#### 3.2.1 Definitions

We modify the definition to restrict the distribution $D$ to be the uniform distribution $U$. That is, we assume that each $x_i \in \{0, 1\}^n$ occurs both in training and testing with probability $2^{-n}$. With this restriction, we can greatly improve analytic tractability and arrive at some very fruitful results.

In this new setting, our definition of a PAC Learning Algorithm is modified as follows:

**Definition 5.** (Uniform Distribution PAC Learning Algorithm) $A$ is a Uniform Distribution PAC Learning Algorithm for a concept class $C$ using a hypothesis class $H$ if for every $\epsilon, \delta > 0$ and every $c \in C$, if $A$ is given access to examples $\langle x, c(x) \rangle$, where $x$ is drawn uniformly from an example space $X$, then with probability at least $1 - \delta$, $A$ outputs $h \in H$ such that $P^U(h(x) \neq c(x)) \leq \epsilon$, where $P^U$ represents probability w.r.t. the uniform measure.

Since each $x_i$ occurs with probability $1/2^n$, we let $X' = \{x \in \{0, 1\}^n \mid h(x) \neq c(x)\}$ and define

$$err(h, c) := P^U(h(x) \neq c(x)) = \frac{|X'|}{2^n}.$$ 

Informally, $err(h, c)$ is the normalized weight of the symmetric difference of the sets where the functions $h$ and $c$ are non-zero.

#### 3.2.2 Approach under Uniform Distribution Model

Broadly speaking, our approach in this setting will be the following:

- Find polynomials $p : \{0, 1\}^n \to \mathbb{R}$ for which $\mathbb{E}\{(p(x) - c(x))^2\}$ is small.

- For $h(x) := \text{sign}(p(x))$, it will be easy to show that $err(h, c)$ is small when $\mathbb{E}\{(p(x) - c(x))^2\}$ is small.

The resulting algorithms will be fast whenever the polynomial $p$ is of low degree or sparse (i.e., with a small amount of non-zero coefficients).
3.2.3 An Application to Learning Poly(n)-term DNFs

To demonstrate that this restriction to the uniform distribution leads to useful results, we return to the problem of learning Poly(n)-term DNFs. Recall that the lower bound of the PTF degree for of class is $\Omega(n^{1/3})$. In the general setting, we have a result from Klivans and Servedio (2004) which gives an algorithm which runs in $n^{O(n^{1/3} \log n)}$ time, which is essentially optimal for the PTF degree method.

However, if we restrict ourselves to the uniform setting, then there exists an algorithm which learns the DNF in $n^{O(\log n)}$ time. To prove this result, the following lemma is needed.

**Lemma 6.** Let $f$ be an $s$-term DNF. Let $g$ be $f$ with long terms of width $\geq \log(s/\tau)$ removed. Then $err_U(f, g) \leq \tau$.

**Proof.** Under the uniform distribution, a term $T$ in $f$ of width $k$ is satisfied with probability $2^{-k}$. Hence a long term is satisfied with probability $2^{-\log(s/\tau)} = \tau/s$. In the worst case, all $s$ terms are long and we have $err(f, g) \leq s \cdot \tau/s = \tau$. \hfill $\square$

Note that in the above proof, the assumption that we were working under $U$ was essential. We now provide a sketch of the proof for learning a $Poly(n)$-term DNF $f$ in $n^{O(\log n)}$ time.

- From the previous lemma, we let $g$ denote $f$ with all terms width at most $\log(Poly(n)/\epsilon)$. Then $err(f, g) \leq \epsilon$.

- View the terms of $g$ as $N = n^{O(\log(Poly(n)/\epsilon))} = n^{O(\log(n/\epsilon))}$ meta-variables, and $g$ as an OR of these meta-variables.

- Run standard algorithm (e.g., Winnow) to learn $f$ in time $n^{O(\log(n/\epsilon))}$.

3.3 Fourier Analysis of Boolean Functions

Boolean functions are by nature discrete, combinatorial objects. However, it is helpful to think of them as a subset of a larger group of functionals (real-valued functions) and apply standard analytical techniques to gain additional information about them.

In this section, we present one such application of the technique of fourier decomposition. Fourier analysis aims to decompose functions into linear combinations of simpler basis functions. The approach is analogous to decomposing vectors in $\mathbb{R}^n$ into linear
combinations of axis variables $x_1, ..., x_n$. In fact, we will often use the words *vector* and *function* interchangeably.

The key points here are:

- For $n \in \mathbb{N}$, the set $V_n := \{ f \mid f : \{-1, 1\}^n \to \mathbb{R} \}$ is a linear space isomorphic to $\mathbb{R}^{2^n}$.
- We can define an inner product on $V_n$, which will allow us to construct an orthonormal basis for $V_n$.
- Using this orthonormal basis, we can decompose functions as linear combinations of these basis functions.
- We can argue about functions $f \in V_n$ by looking at properties of this decomposition (i.e., Fourier analysis).

We will focus on the first three points in today’s lecture.

### 3.3.1 Functions as Vectors

Let $V_n = \{ f \mid f : \{-1, 1\}^n \to \mathbb{R} \}$ be the set of real-valued functions on the boolean $n$-cube. Then $V_n$ is a vector space with function (vector) addition and scalar multiplication defined in the natural way:

$$(f + g)(x) = f(x) + g(x); \quad (cf)(x) = c \cdot f(x).$$

In fact, we can identify every $f \in V_n$ with a unique vector $v_f \in \mathbb{R}^{2^n}$. To do this, first fix an order $x_1, ..., x_{2^n}$ of the elements $x_i \in \{-1, 1\}^n$ and then define the mapping $f \mapsto v_f$ as

$$v_f := (f(x^{(1)}), f(x^{(2)}), ..., f(x^{(2^n)})).$$

This mapping is clearly a bijection. For any $f, g \in V_n$ and $c \in \mathbb{R}$ we have

$$v_{f+g} = v_f + v_g \quad v_{cf} = cv_f.$$

Hence the mapping $f \mapsto v_f$ is a linear transformation and $V_n \cong \mathbb{R}^{2^n}$ are isomorphic vector spaces. So functions on the boolean cube can be thought of simply as vectors in $\mathbb{R}^{2^n}$. In the case of a Boolean function, the interpretation of this mapping is simple. The vector corresponding to a boolean function is the right-side of its truth table.

**Example 7.** Consider the function $f \in V_2$ given by $f(x_1, x_2) = \min(x_1, x_2)$. Then we have the truth table
3.3.2 Basis Functions

Recall from linear algebra that the dimension of a finite-dimensional real vector space $V$, denoted $\dim(V)$, is the cardinality of any maximal set of linearly-independent vectors (there are, in general, many such sets). If $B = \{ e_1, \ldots, e_k \} \subset V$ is any set of linearly-independent vectors such that $k = \dim(V)$, then $B$ forms a basis of $V$ and we may write any vector $v \in V$ as a unique linear combination of basis vectors $e_1, \ldots, e_k$:

$$v = \sum_{i=1}^{k} \lambda_i e_i \quad (\lambda_i \in \mathbb{R}).$$

In the previous section we showed that $V_n \cong \mathbb{R}^{2^n}$ as a vector space. From this and what was said above about dimension, we see that any basis of $V_n$ consists of exactly $2^n$ functions. The simplest and most natural basis is the set of indicator functions for the elements of $\{-1, 1\}^n$.

**Example 8.** Let $\{\delta_i\}_{1 \leq i \leq 2^n}$ be the set of functions in $V_n$ defined by

$$\delta_i(x) = \begin{cases} 1 & x = x^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

Every function $f \in V_n$ can be decomposed as a linear combination of these functions. For instance:

$$\min(x_1, x_2) = -1 \cdot \delta_{\{-1,-1\}} - 1 \cdot \delta_{\{-1,1\}} - 1 \cdot \delta_{\{1,-1\}} + 1 \cdot \delta_{\{1,1\}}.$$

In fact, these basis functions correspond to the standard basis functions when mapped into $\mathbb{R}^{2^n}$ by the mapping $f \mapsto v_f$ defined above.

In order to use the tools of fourier methods, we need to add some more structure to $V_n$. In particular, we need the geometric notion of orthogonality (i.e., the notion of perpendicular vectors), which requires defining an inner product on $V_n$.
Definition 9 (Inner Product of $V_n$). Let $f, g \in V_n$. We define the inner product on $V_n$ to be 
\[ \langle f, g \rangle := \mathbb{E}\{f(x)g(x)\} = \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)g(x). \]

This is merely a generalization of the familiar dot-product from vector calculus, and we shall see that many of the usual properties of the dot-product still apply. In particular,

- Two functions $f, g \in V_n$ are orthogonal (i.e., perpendicular) if and only if $\langle f, g \rangle = 0$ (this can be taken as the definition of orthogonal)
- The norm of a function $f$, a notion of length, is given by $\|f\| = \sqrt{\langle f, f \rangle}$
- If $\|f\| = 1$ then we say that $f$ is a unit vector and $\langle g, f \rangle \cdot f$ is the projection of $g$ along $f$ for any function $g \in V_n$

The last point above says that if we have an orthonormal basis $\mathcal{B} = e_1, \ldots, e_{2^n}$ of $V_n$, then we can write any function $f$ as

\[ f(x) = \sum_{i=1}^{2^n} \langle f, e_i \rangle \cdot e_i \]

It is worth pointing out that in the definition of inner product, the first equality may be taken as a general equality but the second equality is true if and only if we are working in the Uniform Distribution Setting.

3.3.3 Parity Basis Functions

It turns out that a large part of finding useful applications of fourier analysis in any setting is the art of choosing a suitable set of basis vectors (functions). In our case, we will want to consider the set of all parity functions.

Definition 10. Let $S \subseteq \{1, \ldots, n\} := [n]$. The Parity-on-$S$ is defined as

\[ \chi_S := \prod_{i \in S} x_i. \]

We take $\chi_\emptyset = 1$ for reasons we shall see shortly.
From the definition we see that \( \mathbb{E}\{\chi_\emptyset\} = 1 \). If \( S \neq \emptyset \), then \( j \in S \) for some \( j \in [n] \), and we have
\[
\mathbb{E}\{\chi_S\} = \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} \chi_S(x) = \frac{1}{2^n} \left( \sum_{x_j = 1} \chi_{S \setminus \{j\}}(x) - \sum_{x_j = -1} \chi_{S \setminus \{j\}}(x) \right) = \frac{1}{2^n} \cdot 0 = 0.
\]

Claim 11. The parity functions are orthonormal:
\[
\langle \chi_S, \chi_T \rangle = \begin{cases} 
1 & S = T \\
0 & S \neq T 
\end{cases}
\]

Proof. Let \( S \Delta T \) denote the symmetric difference of sets \( S \) and \( T \). Then for \( S, T \subset \{1, \ldots, n\} \) we have
\[
\chi_S(x) \chi_T(x) = \prod_{i \in S} x_i \prod_{j \in T} x_j = \prod_{i \in S \Delta T} x_i \prod_{j \in S \cap T} x_j^2 = \prod_{i \in S \Delta T} x_i = \chi_{S \Delta T}.
\]
Hence
\[
\langle \chi_S, \chi_S \rangle = \mathbb{E}\{\chi_S \chi_S\} = \mathbb{E}\{\chi_{S \Delta S}\} = \mathbb{E}\{\chi_\emptyset\} = 1,
\]
and if \( S \neq T \)
\[
\langle \chi_S, \chi_T \rangle = \mathbb{E}\{\chi_S \chi_T\} = \mathbb{E}\{\chi_{S \Delta T}\} = 0.
\]

Theorem 12. Let \( f \in V \). Then \( f \) has a unique representation as a linear combination of \( \{\chi_S\} \):
\[
f(x) = \sum_{S \subset [n]} \hat{f}(S) \chi_S(x).
\]

The above theorem follows directly from the fact that orthogonal vectors are necessarily linearly independent and that \( |\{\chi_S\}| = 2^n \). So that \( \{\chi_S\} \) is indeed a basis for \( V \).

NOTE: From now on, when we speak of \( \hat{f}(S) \) we will mean with respect to the parity basis functions \( \chi_S \).

3.3.4 Computing the fourier coefficients \( \hat{f}(S) \)

The fourier coefficient \( \hat{f}(S) \) is simply the projection of \( f \) onto the basis function \( \chi_S \):
\[
\langle f, \chi_S \rangle = \left\langle \sum_{T \in [n]} \hat{f}(T) \chi_T, \chi_S \right\rangle = \sum_{T \in [n]} \hat{f}(T) \langle \chi_T, \chi_S \rangle = \hat{f}(S).
\]
The second equality is justified since the inner product is a bi-linear form. For boolean functions, there is an additional interpretation:

\[
\hat{f}(S) = \mathbb{E}\{f(x)\chi_S(x)\} = \mathbb{P}(f(x) = \chi_S(x)) - \mathbb{P}(f(x) \neq \chi_S(x)) = 2\mathbb{P}(f(x) = \chi_S(x)) - 1.
\]

So we can view \(\hat{f}(S)\) as a measure of correlation between \(f\) and \(\chi_S\).

### 3.3.5 An Example of Fourier Decomposition

Consider the function

\[
f(x) = \text{AND}(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if all } x_i = -1 \\ 0 & \text{otherwise} \end{cases}
\]

Then we can write

\[
f(x) = \prod_{i=1}^{n} \left(\frac{1 - x_i}{2}\right)
\]

So for a given \(S \subseteq [n]\) we compute

\[
\hat{f}(S) = \mathbb{E}\left\{\prod_{i\in[n]} \frac{1 - x_i}{2} \prod_{j\in S} x_j\right\}
\]

\[
= \mathbb{E}\left\{\prod_{i\notin S} \frac{1 - x_i}{2} \prod_{j\in S} \frac{x_j - 1}{2}\right\}
\]

\[
= -1^{|S|} \mathbb{E}\left\{\prod_{i\notin S} \frac{1 - x_i}{2} \prod_{j\in S} \frac{1 - x_j}{2}\right\}
\]

\[
= -1^{|S|} \mathbb{E}\{f\} = -\frac{1^{|S|}}{2^n}.
\]

We again note that the expectation here is taken over a uniform distribution. Hence the unique fourier decomposition of \(\text{AND}(x)\) is given by

\[
\text{AND}(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S = \sum_{S \subseteq [n]} -\frac{1^{|S|}}{2^n}\chi_S
\]
3.3.6 Decision Trees

As a final example we consider the function \( f(x) \) given by the following decision tree \( T \):

![Decision Tree Diagram]

Let \( p(x) \) be the path of the tree associated with a term \( x \in \{0, 1\}^n \) and \( f(p) \) denote the value at the leaf of path \( p \). Then we can write

\[
f(x) = \sum_{p \in T} 1_{p(x)} \cdot f(p),
\]

where the sum is taken over all paths in the tree \( T \), and \( 1_{p(x)} \) is the indicator function for the path in the tree.

Each term \( 1_{p(x)} f(p) \) is a polynomial of degree equal to the length of the path. For instance, the path marked with bullets above is represented by the term:

\[
\frac{1}{23}(1 - x_1)(1 + x_2)(1 - x_3).
\]

It follows that \( f \) is a degree 3 polynomial. Since \( f(x) = \sum_{S \subseteq [4]} \hat{f}(S) \chi_S \), we know that the only basis functions in the fourier representation of this decision tree are those with \( |S| \leq 3 \). In particular, \( \hat{f}(1, 2, 3, 4) = 0 \).