Lecture 2: Fourier expansion

Rajat Mittal

IIT Kanpur

We saw definition, different representations and examples of Boolean functions. It was also mentioned that there exists a polynomial representation for a Boolean function $f : \{0,1\}^n \to \{0,1\}$. We will see that any function $f : \{0,1\}^n \to \mathbb{R}$ can be represented as a real polynomial over n variables. Notice that the range is generalized to $\mathbb{R}$.

Remember that the Boolean domain can also be thought of as $\{-1,1\}$. The polynomial representation of a function $f : \{-1,1\}^n \to \mathbb{R}$ is called the *Fourier expansion* of the function and it is one of the main tools to analyze Boolean functions. This lecture note will introduce you to *the Fourier analysis of Boolean functions*. That means, we will define the Fourier expansion and look at some of the basic properties.

An excellent reference for this topic is the book *Analysis of Boolean functions* by Ryan O'Donnell [1]. Much of the content of this lecture note is inspired from that book.

1 Polynomial representation of Boolean functions

Remember, the easiest way to specify a Boolean function is by its truth table. That means, a function $f : \{0,1\}^n \to \mathbb{R}$ is represented by 2^n real numbers. In other words, a function $\{0,1\}^n \to \mathbb{R}$ can be represented as a vector in vector space $\mathbb{R}^{2^n}$.

You already know a lot of examples of these Boolean functions. Notice that some of the functions have range as real numbers.

- OR: It is 0 if and only if all inputs are 0.
- AND: It is 1 if and only if all inputs are -1.
- NOR: It is 1 if and only if all inputs are 0.
- PARITY: It is easier to define this function as PARITY : $\{-1,1\}^n \to \{-1,1\}$. With this domain and range, it is simply the multiplication of all the inputs, $\text{PARITY}(x) = x_1 x_2 \cdots x_n$.
- Constant function: Its value is $c \in \mathbb{R}$ for all inputs.

Exercise 1. What would be the Majority function?

Let us consider all functions of type $f : \{0,1\}^n \to \mathbb{R}$. We saw that PARITY and constant function can be represented as a polynomial in variables. Is it possible to represent all Boolean functions as polynomials?

Exercise 2. Can you represent all functions $f : \mathbb{R} \to \mathbb{R}$ as polynomials?

The surprising answer in this case is, YES!! The trick is called *interpolation*.

Interpolation:

Exercise 3. Can you find a polynomial which is 1 on all 1's input and 0 otherwise? What is this function called?

The answer to the previous exercise is not very difficult, and it is the polynomial $x_1 x_2 \cdots x_n$.

Note 1. It is the same polynomial as PARITY, but the domains are different.

The same technique can be generalized for an arbitrary input (not just all 1's input).

Exercise 4. Fix an input a, can you find a polynomial which is 1 on a and 0 otherwise.

These polynomials are called *indicator polynomials*. Using these polynomials, every function $f : \{0,1\}^n \to \mathbb{R}$ can be written as a polynomial.

$$f(x) = \sum_{a \in \{0,1\}^n} f(a) \prod_{i:a_i=1} x_i \prod_{i:a_i=0} (1 - x_i).$$

Exercise 5. Verify that the polynomial defined above agrees with the function value on every input $x \in \{0,1\}^n$. Notice that the polynomial can take any input from $\mathbb{R}^n$.

Observe that these polynomials are of special kind, every variable appears at most once in any monomial. This is possible, because $x_i^2 = x_i$ for this domain. Such multivariate polynomials are called *multi-linear polynomials*. We have shown that every Boolean function can be represented as a multi-linear polynomial. Converse is trivially true. You will show in the assignment that this representation is unique.

Let $\mathbb{1}_a := \prod_{i:a_i=1} x_i \prod_{i:a_i=0}(1 - x_i)$ denote the indicator polynomial which is 1 at a and 0 otherwise. Then,

$$f(x) = \sum_{a \in \{0,1\}^n} f(a)\mathbb{1}_a.$$

Exercise 6. What will be the polynomial representation of *not all equal* (NAE) function on three bits, where the function value is 1 if and only if all the three bits are not same.

2 Fourier representation

We have defined a Boolean function $f : \{0,1\}^n \to \{0,1\}$ on $\{0,1\}$ domain. We can switch to $\{-1,1\}$ domain, by replacing 0 with 1 and 1 with -1. So, a Boolean function can also be defined as a function from $\{-1,1\}^n$ to $\{-1,1\}$. Generalizing, a function with Boolean domain, $f : \{-1,1\}^n \to \mathbb{R}$ can be viewed as a vector in $\mathbb{R}^{2^n}$. Conversely, any element of $\mathbb{R}^{2^n}$ can also be thought of as a function from $\{-1,1\}^n$ to $\mathbb{R}$. In other words, the space of all such functions, from $\{-1,1\}^n$ to $\mathbb{R}$, is same (isomorphic to) as $\mathbb{R}^{2^n}$.

Our goal is to represent every function of the form $f : \{-1,1\}^n \to \mathbb{R}$ as a polynomial. That will be called the *Fourier expansion* of the function f. We can apply the same trick as $\{0,1\}$ domain, we only need to know the indicator polynomials for inputs $a \in \{-1,1\}^n$.

Exercise 7. What is the indicator polynomial for $a \in \{-1,1\}^n$?

Using the linear transformation to transfer $\{0,1\}$ domain to $\{-1,1\}$ domain,

$$\mathbb{1}_a = \prod_{i:a_1=1} \frac{1 + x_i}{2} \prod_{i:a_1=-1} \frac{1 - x_i}{2}.$$

Hence, any function $f : \{-1,1\}^n \to \mathbb{R}$ can be written as

$$f(x) = \sum_{a \in \{-1,1\}^n} f(a)\mathbb{1}_a.$$

Writing it in terms of monomials (notice that the degree of every variable is at most 1 in every variable),

$$f(x) = \sum_{s \in \{0,1\}^n} \hat{f}(s) \prod_{s_i=1} x_i.$$

A string $s \in \{0,1\}^n$ can be identified uniquely with a subset of $[n]$.

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x) \qquad (\textit{here } \chi_S(x) = \prod_{i \in S} x_i) \tag{1}$$

This is called a *Fourier representation* or *Fourier expansion* of f. The coefficients, $\hat{f}(S)$'s, are called the Fourier coefficients of f.

Note 2. Fourier expansion has been defined for the range $\mathbb{R}$, that means the function f takes $\{-1,1\}^n$ as input and outputs a real number. The expansion will have *special* properties when the range is restricted to $\{-1,1\}$. These properties will be useful later.

Let us write the Fourier representation of AND function on two bits. In $\{-1,1\}$ domain, AND is -1 iff all inputs are -1. This implies,

$$\text{AND}(x_1,x_2) = (-1)\mathbb{1}_{-1,-1} + \mathbb{1}_{-1,1} + \mathbb{1}_{1,-1} + \mathbb{1}_{1,1}.$$

Using the definition of indicator functions in this domain,

$$\text{AND}(x_1,x_2) = \frac{1}{4}\left((-1)(1-x_1)(1-x_2) + (1-x_1)(1+x_2) + (1+x_1)(1-x_2) + (1+x_1)(1+x_2)\right).$$

This can simplified to give the Fourier representation of AND on two bits,

$$\text{AND}(x_1,x_2) = \frac{1}{2}\left(1 + x_1 + x_2 - x_1x_2\right).$$

Exercise 8. Write the Fourier representation of MAJORITY function on 3 bits.

We have represented functions on Boolean domain in terms of Boolean functions $\chi_S(x) := \prod_{i\in S} x_i$. They can be thought of as *partial parities*, parities on the subset S.

Exercise 9. Show that χ_S are Boolean functions.

In other words, these χ_S, for different S, form a basis of $\mathbb{R}^{2^n}$. As a sanity check, you can see that the dimension is 2^n and number of partial parities is also 2^n. Actually, these partial parities for different subsets $S \subseteq [n]$ are orthogonal to each other, we will prove it soon.

We can define a normalized inner product over the space of all functions from $\{-1,1\}^n$ to $\mathbb{R}$ (same as $\mathbb{R}^{2^n}$),

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

For $\mathbb{R}^{2^n}$, generally the inner product is defined as $\sum_{x\in\{-1,1\}^n} f(x)g(x)$. The difference is a normalization factor of $\frac{1}{2^n}$. The normalization factor will make the basis of partial parties orthonormal, and not just orthogonal.

Theorem 1. *The partial parities, χ_S, form an orthonormal basis of $\mathbb{R}^{2^n}$.*

Proof. We will show that for two distinct subsets S, T, $\langle\chi_S|\chi_T\rangle = 0$. This will show that χ_S's form an orthogonal basis, you can show that it is orthonormal too (use the normalized inner product define above).

First, observe that the inner product can be simplified, $\langle\chi_S|\chi_T\rangle = \sum_{x\in\{-1,1\}^n} \chi_{S\triangle T}(x)$. Here, $S\triangle T$ is the symmetric difference between two sets. This observation follows from the polynomial representation of χ_S and χ_T.

What is the sum of outputs of a partial parity? The quantity $\sum_{x\in\{-1,1\}^n} \chi_S(x)$ is 0 except iff $S = \varphi$; you will prove this in the assignment. This shows that $\langle\chi_S|\chi_T\rangle$ is 0 except if $S = T$ (What is $\langle\chi_S|\chi_S\rangle$?). Hence, χ_S and χ_T are orthogonal if $S \neq T$.

Noticing that $\langle\chi_S|\chi_S\rangle = 1$, we get that these partial parities, χ_S's, form an orthonormal basis. □

This shows that functions $\chi_S(x)$ form an orthogonal basis of all multi-linear polynomials. This also implies that Fourier representation is unique. Every function can be uniquely expressed in the form given by Eqn. 1.

Fourier degree of a function: One possible complexity measure of these Boolean functions could be the degree of a function (also called *Fourier degree*). We can define degree of a monomial as sum of degrees of variables. Then, degree of a multivariate polynomial is the maximum degree of a monomial in its unique representation.

A function with low degree would be simple and high degree will be considered complicated.

Exercise 10. What is the maximum possible degree of a Boolean function on n variables?

Constant functions have 0 degree and are the simplest functions. Functions of kind $x_1, x_1 + x_2, x_1 + x_2 + \cdots + x_n$ are simple under this measure and have degree 1. Parity would be a complex function with maximum degree n.

3 Fourier analysis over a finite Abelian group (extra reading)

You might have heard of *Fourier transform* as the conversion of a function from time domain to frequency domain. It has applications in analysis of differential equations, spectroscopy, quantum mechanics and signal processing.

Instead, we will introduce Fourier transforms as a general tool to analyze functions over Abelian groups (mostly finite). In general, Fourier transform acts on functions over these Abelian groups (e.g., real or complex numbers) and gives representation of those functions in the *character basis*. Readers are encouraged to look at the definitions of character, group theory and Fourier analysis over an Abelian group. The next few paragraphs will use facts from the basics of character theory and can be found in any standard text on group theory.

Given a function $f : G \to \mathbb{C}$ on an *Abelian group* G, there is a standard representation of the function as a $\mathbb{C}^{|G|}$ vector.

Note 3. If you are confused about a general group, take G to be $\mathbb{Z}_n$, the group of remainders modulo n.

Exercise 11. What is this representation?

The representation is simply the value of the function on different elements of G, i.e., as a vector in $\mathbb{C}^{|G|}$.

Let us consider some more interesting functions of the kind $\chi : G \to \mathbb{C}$. A *character* χ of G is the non-zero function $\chi : G \to \mathbb{C}$, s.t.,

$$\chi(g_1)\chi(g_2) = \chi(g_1 g_2) \quad \forall g_1, g_2 \in G$$

Exercise 12. Can you think of a simple function which is a character?

The function $\chi_0(g) = 1$, for all $g \in G$, is known as the trivial character and is denoted by χ_0.

Exercise 13. What is $\chi(e)$, where e is the identity element?

For any element g in a finite group G, there exists an n such that $g^n = e$, where e is the identity element. You can show (as an exercise) that $|\chi(g)| = 1$, for any g in G.

Exercise 14. What are the characters for $\mathbb{Z}_2$?

Properties of characters: These properties of characters are well-known. You are encouraged to read any standard text on group representations for proofs (or try them yourselves). Below, χ will denote a character and G is a finite Abelian group. Observe that a character χ can be viewed as a vector over complex numbers with $|G|$ length. With an abuse of notation, let χ also denote the normalized vector corresponding to function χ.

- For a non-trivial character χ, $\sum_{g \in G} \chi(g) = 0$. If χ is trivial, then this sum is $|G|$.
- The product of two characters is a character.
- Any two distinct characters of G are orthogonal to each other (as vectors in $\mathbb{C}^{|G|}$).

– There are exactly $|G|$ many distinct characters for G (this is harder to prove).
– With appropriate normalization, the characters of G form an orthonormal basis of the space $\mathbb{C}^{|G|}$.

Note 4. It is known that the set of characters of G form a group under multiplication, called $\hat{G}$. We already saw that the size of $\hat{G}$ and G is same, not just that, they are actually isomorphic. In other words, we can index characters of G with elements of G (though this indexing is not unique).

Using these properties, we can define *the Fourier transform* over an Abelian group G. The Fourier transform is basically the *basis change operator* from the standard basis $e_1, e_2, \cdots, e_{|G|}$ to the *character basis* $\chi_1, \chi_2, \cdots, \chi_{|G|}$.

Any function f in $\mathbb{C}^{|G|}$ can be represented in Fourier basis as,

$$f = \sum_{i=1}^{|G|} \hat{f}(i)\chi_i.$$

The coefficients $\hat{f}(i)$ is called the i-th Fourier coefficient of f. Taking inner product with χ_i,

$$\hat{f}(i) = \langle \chi_i | f \rangle.$$

Exercise 15. Why is the character basis different from any other orthonormal basis?

Hint: Look at the entry-wise multiplication of character basis vectors.

These Fourier coefficients capture properties of our function which might not be evident in the standard representation. For instance, let us look at the first Fourier coefficient, one corresponding to the trivial representation. By definition,

$$\hat{f}(1) = \langle \chi_1 | f \rangle = \frac{1}{|G|} \sum_x f(x). \tag{2}$$

In other words, first Fourier coefficient tells us about the summation of function value at all points.

4 Properties of Fourier expansion

To summarize the findings of previous section, any function $f : \{-1, 1\}^n \to \mathbb{R}$ can be written as,

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x).$$

This is called the Fourier expansion of f, and $\hat{f}(S)$ are called the Fourier coefficients. We know that the functions χ_S's form an orthonormal basis of the vector space of all functions from $\{-1, 1\}^n$ to $\mathbb{R}$. The perspective of partial parities as orthonormal basis allows us to find the Fourier coefficients easily. Taking the inner product of Fourier expansion with χ_S,

$$\hat{f}(S) = \langle \chi_S | f \rangle.$$

Exercise 16. Find the Fourier coefficients of AND on two bits using this formula, check with the Fourier representation computed earlier.

Notice that the Fourier coefficients of same degree (monomials with same size) have same value for AND function. You will show in the assignment that for any symmetric function, the Fourier coefficients at the same level are equal.

We have considered two orthonormal bases to represent these functions. The truth table or vector representation is the expansion in terms of standard basis, with basis elements $\mathbb{1}_a$ for all $a \in \{-1, 1\}^n$. Fourier expansion is the representation in the Fourier basis, where each basis element is χ_S for some $S \subseteq [n]$.

The matrix which takes a function from standard basis to Fourier basis is called the Fourier transform over $\mathbb{Z}_2^n$. You will find the entries of this matrix in the assignment.

This basis change interpretation allows us to prove some basic but important results. First, the inner product between two vectors should not change.

Theorem 2 (Plancherel's Theorem). *For any two f, g from $\{-1,1\}^n$ to $\mathbb{R}$,*

$$\langle f|g\rangle = \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S).$$

If you are confused about the normalization factor, consider the usual inner product (without normalization), χ_S's need to be normalized by $\frac{1}{\sqrt{2^n}}$ to make them orthonormal. This implies that the *coordinates* corresponding to these normalized χ_S's will be $\hat{f}(S)$ multiplied by $\sqrt{2^n}$. This will give the equation,

$$\sum_{x\in\{-1,1\}^n} f(x)g(x) = 2^n\langle f|g\rangle = \sum_{S\subseteq[n]} (\sqrt{2^n}\hat{f}(S))(\sqrt{2^n}\hat{g}(S)).$$

It is easy to see that this equation is same as Plancherel's theorem.

For $f = g$, Plancherel's theorem will give us,

$$\sum_{S\subseteq[n]} \hat{f}(S)^2 = \frac{1}{2^n}\sum_{x\in\{-1,1\}^n} f(x)^2. \tag{3}$$

For a Boolean function $\frac{1}{2^n}\sum_{x\in\{-1,1\}^n} f(x)^2 = 1$, giving the very important *Parseval's identity.*

Theorem 3 (Parseval's identity). *For any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$,*

$$\sum_{S\subseteq[n]} \hat{f}(S)^2 = 1.$$

Note 5. The equation above and Eq. 3, both are known as Parseval's identity.

Uniform probability distribution over inputs: An important probability distribution, for many applications, is the uniform distribution over all inputs $x \in \{-1,1\}^n$. When not mentioned, this will be the default distribution in our discussion. Under this distribution,

$$\mathrm{E}[f(x)] = \frac{1}{2^n}\sum_{x\in\{-1,1\}^n} f(x) = \hat{f}(\varphi).$$

Here, φ is the empty set. This follows from linearity of expectation and $\mathrm{E}[\chi_S(x)] = 0$ iff $S \neq \varphi$.

Exercise 17. Show that Plancherel's theorem can be expressed as,

$$\mathrm{E}[f(x)g(x)] = \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S).$$

Like expectation (treating $f(x)$ as a random variable), we can talk about variance too.

$$\mathrm{Var}(f) = \mathrm{E}[(f(x) - \mathrm{E}[f])^2] = \mathrm{E}[f(x)^2] - \mathrm{E}[f]^2.$$

Using Parseval's identity,

$$\mathrm{Var}(f) = \sum_{S\subseteq[n], S\neq\varphi} \hat{f}(S)^2.$$

Convolution: Convolution is an important operator over two functions. The convolution of two functions $f, g : \{-1,1\}^n \to \mathbb{R}$ is defined as,

$$f * g(x) = \mathrm{E}_y[f(y)g(xy)].$$

Here, xy denote the entry-wise multiplication of two strings x and y. Notice that the expectation is taken over y which comes from the uniform probability distribution over inputs (x is fixed and y is a random variable).

It turns out that it is easy to write the Fourier expansion of convolution of two functions.

Theorem 4. *Let $f * g$ denote the convolution of two functions $f, g : \{-1,1\}^n \to \mathbb{R}$, then*

$$\widehat{f * g}(S) = \hat{f}(S)\hat{g}(S).$$

Proof. The theorem follows from direct manipulation of Fourier formulas.

$$\begin{aligned}
\widehat{f * g}(S) &= \langle f * g | \chi_S \rangle \\
&= \frac{1}{2^n} \sum_x \chi_S(x) \mathrm{E}[f(y)g(xy)] \\
&= \frac{1}{2^{2n}} \sum_{x,y} \chi_S(xy)\chi_S(y) f(y) g(xy) \\
&= \frac{1}{2^{2n}} \sum_y f(y)\chi_S(y) (\sum_x g(xy)\chi_S(xy)) \\
&= \frac{1}{2^n} \hat{g}(S) \sum_y f(y)\chi_S(y) \quad \textit{(for a fixed } y\textit{, } xy \textit{ goes over all strings)} \\
&= \hat{f}(S)\hat{g}(S)
\end{aligned}$$

□

The number $\hat{f}(S)^2$ is called the Fourier weight of S for function f. They can be used to define a probability distribution over subsets of $[n]$.

Note 6. Even though the Fourier weights do not contain complete information about f (two different functions can have same Fourier weights), they are very useful in many situations.

One way to summarize these properties is: we take a formula/equation/estimate over the standard basis and write it in terms of Fourier basis. This allows us to write many properties of Boolean functions in terms of their Fourier coefficients. This alternate representation is useful in many applications, we will see one such application next.

5 Linearity testing

This section is taken from the book *Analysis of Boolean functions* by Ryan O'Donnell [1].

Property testing is an area of theoretical computer science, where we want to check whether a combinatorial object (graph, functions) has a global property or not, using queries to the object. To test the property exactly, the number of queries will be very high. The aim of property testing is to test the property using very few queries, with the compromise that the object either satisfies the property or is *far away* from satisfying the property.

One of the landmark result in this field is the linearity testing algorithm by Blum, Luby and Rubinfeld. The task is to find whether a given Boolean function is *linear* or not. A Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ is linear iff $f(xy) = f(x)f(y)$ for all pairs x, y. You will show in the assignment that partial parities (χ_S) are linear and they are the only linear functions.

In the context of property testing, we want to test whether a Boolean function is linear or not, given

– the oracle to query the function at any x (the cost of the algorithm is measured by the number of queries),
– the promise that the function is linear or *far* from linear. A function is far from linear if for any S, $f(x) \neq \chi_S(x)$ for at least a constant fraction of inputs.

The algorithm to test linearity is going to be pretty simple. We will pick $x, y \in \{-1,1\}^n$ randomly and check whether $f(xy) = f(x)f(y)$ or not (it can be done for constant number of pairs to increase the success probability). If there is any pair for which $f(xy) \neq f(x)f(y)$, we will output that function is not linear (otherwise it is linear). We will show that this constant query algorithm will test whether a function is linear or not with high probability. Notice that to check linearity exactly, we need to query $\Theta(2^n)$ inputs.

Exercise 18. Convince yourself that if the function is linear, we will always output that the function is linear. Also that this algorithm uses only constant number of queries.

The proof that this algorithm works with high probability is equivalent to showing $f(xy) \neq f(x)f(y)$ for a constant fraction of input pairs (given that function is far from linear). Another way to look at this proof is, this is a robust version of equality between the standard definition of linearity ($f(xy) = f(x)f(y)$) and that a linear function is a parity on some subset.

Theorem 5. *Suppose f is a Boolean function such that $f(x) \neq \chi_S(x)$ for at least a constant fraction (say $1/3$) of inputs for all S. Then, for at least a constant fraction ($1/3$) of pairs $x, y \in \{-1,1\}^n$,*

$$f(xy) \neq f(x)f(y).$$

Proof. We know that f is Boolean function, so it outputs from the set $\{-1,1\}$. This implies that for a pair (x,y), $f(xy) \neq f(x)f(y)$ if and only if $f(xy)f(x)f(y) = -1$ (otherwise $f(xy)f(x)f(y) = 1$). In other words, we are interested in the expectation of $\frac{1}{2}(1 + f(x)f(y)f(xy))$, when x, y are picked uniformly from $\{-1,1\}^n$. Let δ be the fraction of pairs where $f(xy) = f(x)f(y)$. (This is the failure probability of our algorithm.)

Analyzing this expectation,

$$\delta = \mathrm{E}[\frac{1}{2}(1 + f(x)f(y)f(xy))] = 1/2 + 1/2\mathrm{E}[f(x)f(y)f(xy)] = 1/2 + 1/2\mathrm{E}_x[f(x)\mathrm{E}_y[f(y)f(xy)]].$$

The term inside the expectation over x can be seen as a product of $f(x)$ and the convolution of f, f.

$$\delta = 1/2 + 1/2\mathrm{E}_x[f(x)(f * f)(x)].$$

Again, we can use Plancherel's theorem,

$$\delta = 1/2 + 1/2\sum_S \hat{f}(S)\widehat{f * f}(S) = 1/2 + 1/2\sum_S \hat{f}(S)^3.$$

For the last equality, we used the property of the Fourier coefficients of the convolution. We can already see the idea, if a function is linear, exactly one of $\hat{f}(S) = 1$. Then, δ will be one. If a function is far from any χ_S, all $\hat{f}(S)$ will be away from 1 and δ will be small. Let us formalize it.

If f and χ_S are *far*, that means $f(x) = \chi_S(x)$ on less than $2/3$ fraction of x's for any S.

Exercise 19. Show that $\hat{f}(S) \leq 1/3$ for all S.

This means, $\delta \leq 1/2 + 1/2\left(\max_S \hat{f}(S)\right)\sum_S \hat{f}(S)^2$. Using Parseval's identity,

$$\delta \leq 1/2 + 1/2 \cdot 1/3 \leq 2/3.$$

We have shown that if f is far from linear than $\delta \leq 2/3$, that means linearity condition will not be satisfied for at least $1/3$ fraction of pairs, proving the theorem. □

Again, let me emphasize the use of Fourier representation. Many natural properties/estimates of Boolean functions can be converted into Fourier domain. It is sometimes easier to prove these properties in Fourier domain. This specifically happens when the string structure of $\{-1,1\}^n$ is needed (like Hamming distance) and not just the vector space structure of $\mathbb{R}^{2^n}$.

6 Hadamard codes

Noise is a major problem in transmitting messages across any communication channel. It disturbs the messages and can make them unreadable. Any practical communication protocol needs to guard its messages from noise. We assume that the noise is random and *not* introduced by any adversary. We can also assume that the message is a string in $\{-1,1\}^n$.

One of the most successful resolution is to introduce redundancy in the message. Since each bit will be changed by noise with some small probability, we can assume that only a few bits are changed by noise (with high probability). Using redundancy, the receiver can correct the errors on her side. For example, if we need to transmit a bit, we can *encode* -1 as $-1-1-1$ and 1 as 111. Assuming that the noise only corrupts at most one bit of the message, simple majority will tell the receiver which bit sender wanted to transmit. Such encodings are called error correcting codes.

Let us look at the framework to describe a classical error correcting code. The list of symbols which can be transmitted over a channel is called the *alphabet* of that channel. For sake of simplicity, we assumed that our alphabet is $\{-1,1\}$. The noise (error in transmission) can flip -1 to 1 or 1 to -1. In general, any set Σ can be an alphabet and an error can take one element of Σ to another element of Σ.

If we need to send l distinct messages over the channel, we choose l different codewords from $\{-1,1\}^k$, such that, we can check if the error has happened in the transmission and potentially correct it. The set of l codewords $c_1, c_2, \cdots, c_l \in \{-1,1\}^k$ is called the error correcting code. To explain the repetition code from the introduction in this framework, there will be two codewords, $(-1-1-1)$ and (111). Notice that the mapping of messages to codewords is not very important; the messages can be thought of as labels for the codewords.

Succinctly, an error correcting code is a list of l codewords, $c_1, c_2, \cdots, c_l \in \{-1,1\}^k$. When is this code good? The code is good if, even after some noise, receiver can guess the codeword sent to her.

How will noise affect our codewords? Given a codeword $c \in \{-1,1\}^k$, it has k bits and we will assume that the noise will only change few of those bits arbitrarily. So, the noise will change a codeword c to some $d \in \{-1,1\}^k$ such that c and d are *close*. Intuitively, we will be able to identify the error, if for any pair (i,j), the codewords c_i and c_j are *far* from each other.

Exercise 20. What do we mean by close/far in the sentence above?

We will use *Hamming distance* as the notion of distance between two codewords. Just to remind you, Hamming distance between two codewords c_i, c_j is the number of co-ordinates where they differ. From the definitions, it is easy to see the following theorem,

Theorem 6. *If the minimum Hamming distance between any two codewords is d, then we can detect error in transmission if the noise affects at most $d-1$ co-ordinates.*

Exercise 21. Prove the theorem.

The minimum Hamming distance between two codewords (d in the theorem above) is called the *distance* of the code. Intuitively, we would want to have codes with large distance. At this point we should clarify the difference between detecting error and correcting error. Even though we can detect error of up to $d-1$ bits using a code with distance d, we can correct errors with this code only if there are less than $d/2$ bit errors. Why?

One of the interesting error correcting code with large distance is called the *Hadamard code*. It encodes a binary string $S \in \{-1,1\}^n$ (think of it as an indicator for a subset $S \subseteq [n]$) by a code of length $N = 2^n$, by concatenating $\chi_S(x)$ for every x (you can fix any ordering of x, say lexicographic).

Lemma 1. *The distance of the Hadamard code with length $N = 2^n$ is $N/2$.*

Proof. The distance between two codewords, S and T, is the number of x's such that $\chi_S(x) \neq \chi_T(x)$. We already saw that χ_S and χ_T are orthogonal to each other.

$$2^n \langle \chi_S | \chi_T \rangle = |\{x : \chi_S(x) = \chi_T(x)\}| - |\{x : \chi_S(x) \neq \chi_T(x)\}|.$$

Since $\langle \chi_S | \chi_T \rangle = 0$, this implies $|\{x : \chi_S(x) \neq \chi_T(x)\}| = |\{x : \chi_S(x) = \chi_T(x)\}| = N/2$, proving the result. Observe that, not only the minimum distance between any two codewords, but distance between any two codewords is exactly $N/2$ for Hadamard code. □

In other words, if the number of errors are less than $N/4$, we can uniquely decode the received word (some string in $\{-1,1\}^N$). There is an easy algorithm for decoding, view the received word $w \in \{-1,1\}^N$ as a function on $\{-1,1\}^n$.

Exercise 22. Can you guess the algorithm?

You need to simply output S for which $\hat{w}(S)$ is maximum. Suppose the original codeword is S_0. Since the correct codeword $\chi_{S_0}(x)$ agrees with w on at least $3N/4$ places, $\hat{w}(S_0) = \langle w | \chi_{S_0} \rangle > 1/2$ (why?).

Exercise 23. Can you show that there can't be more than three S_i's such that $\hat{w}(S_i) > 1/2$?

We will show a stronger result, there can be no other T (except S_0) such that $\hat{w}(T) > 1/2$. Suppose T is such a set, by a similar argument, χ_T and w will differ on less than $N/4$ places. This will imply that S_0, T differ on less than $N/2$ places, giving a contradiction.

6.1 List decoding of Hadamard codes

We have seen that an error correcting code with distance d can correct errors up to $d/2$ bits. In other words, given that the received word differs from the original codeword by distance at most $d/2$, it can be uniquely decoded.

Exercise 24. Can you improve the correction distance to more than $d/2$?

Since, the correction threshold of $d/2$ cannot be improved, what if the errors are more than $d/2$. One relaxation is called *list decoding*, and the idea is to give a list (small) of codewords which could have been potential original codewords, even if the errors are large (more than $d/2$).

It turns out that Hadamard code has strong list decoding properties. We will show that if number of errors are smaller than $N(\frac{1}{2} - \epsilon)$ (think of ϵ to be a very small constant), there are only a small number of possible codewords.

If w is the received word, the first step is to relate potential codewords with $\hat{w}(S)$'s. Like before, if w and χ_S differ by at most $N(\frac{1}{2} - \epsilon)$ bits, then $\hat{w}(S) > 2\epsilon$. This follows from the expression for $\langle w | \chi_S \rangle$.

From Parseval's identity, the sum of square of Fourier coefficients is 1. In other words, there cannot be many big Fourier coefficients. Specifically, there cannot be more than $\frac{1}{4\epsilon^2}$ Fourier coefficients with value greater than or equal to 2ϵ.

This shows that even if the number of errors are as large as $N(\frac{1}{2} - \epsilon)$, there is at most a list of $\frac{1}{4\epsilon^2}$ possible codewords. Important thing to note is, the number of codewords (list size) only depends on ϵ and not N.

Exercise 25. What will be the size of the list when the error is at most $(9/20)N$?

Goldreich and Levin were able to show that this list can be found efficiently (in time polynomial in n).

Acknowledgements: I would like to thank Aditya Ranjan for pointing out errors in these notes.

7 Assignment

Exercise 26. Given a function $f : \{0,1\}^n \to \mathbb{R}$, show that it can't have two different multi-linear representations.

Exercise 27. Show that $\sum_{x \in \{-1,1\}^n} \chi_S(x)$ is 2^n if $S = \varphi$ and 0 otherwise.

Exercise 28. Show that for any function f changing the domain from $\{-1,1\}^n$ to $\{0,1\}^n$ does not affect the degree.

Exercise 29. The matrix which takes a function from standard basis to Fourier basis is called the Fourier transform over $\mathbb{Z}_2^n$. Explicitly find the entries of Fourier transform.

Exercise 30. For any element g in a finite group G, there exists an n such that $g^n = e$, where e is the identity element. Show that $|\chi(g)| = 1$ for any g in G.

Exercise 31. What are the characters of $\mathbb{Z}_n$.

Exercise 32. Show that χ_S is linear for any S. More importantly, show that any linear function is equal to χ_S for some S.

Exercise 33. Show that every codeword in a Hadamard code with length N (except the trivial one) has Hamming weight $N/2$.

Exercise 34. Show that $\hat{f}(S) = \hat{f}(T)$ when $|S| = |T|$ for a symmetric function f.

References

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