

# The Harmonic Analysis of Boolean Functions

Jacob Denson

February 27, 2017

# Table Of Contents

<b>1</b>	<b>Introduction to Boolean Analysis</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Fourier Analysis and Probability Theory . . . . .	11
1.3	Testing Linearity . . . . .	14
1.4	The Walsh-Fourier Transform . . . . .	19
<b>2</b>	<b>Social Choice Functions</b>	<b>23</b>
2.1	Influence . . . . .	26
2.2	Noise and Stability . . . . .	33
2.3	Arrow's Theorem . . . . .	39
<b>3</b>	<b>Spectral Complexity</b>	<b>44</b>
3.1	Spectral Concentration . . . . .	44
3.2	Decision Trees . . . . .	46
3.3	Computational Learning Theory . . . . .	48
3.4	Goldreich-Levin Theorem . . . . .	51
<b>4</b>	<b>Property Testing</b>	<b>55</b>
4.1	Dictator Tests . . . . .	56
4.2	Probabilistically Checkable Proofs . . . . .	57
4.3	Constraint Satisfaction and Property Testing . . . . .	60
4.4	Hastad's Hardness Theorem . . . . .	61

# Chapter 1

## Introduction to Boolean Analysis

### 1.1 Introduction

This course is about the theory of harmonic analysis, and how it applies to discrete, binary structures. The rough idea is that if we are trying to understand some set  $X$  with a ‘binary representation’, in the sense that it can be put into one to one correspondence with  $\{0,1\}^n$ , then the abelian group structure on  $\{0,1\}^n$  induces a natural group structure on  $X$ , and we can obtain powerful results about  $X$  provided the induced group operation is meaningful to the structure of  $X$ . In particular, if we want to obtain quantitative results about a function  $f : X \rightarrow \mathbf{R}$  and its translation  $(\sigma_y f)(x) = f(x + y)$  with respect to the group operation induced on  $X$ , then looking at the Fourier series of  $f$  with respect to the group operation becomes interesting, because the characters of the group simultaneously diagonalize the operators  $\sigma_y$ .

**Example.** *The Hamming distance between two length  $n$  binary strings  $x$  and  $y$  is the number of indices where the strings differ, which we denote by*

$$\Delta(x, y) = \#\{i : x^i \neq y^i\}$$

*Then  $\Delta(x, y)$  is the  $L^1$  norm, viewing elements of  $\{0,1\}^n$  as  $\{0,1\}$  indicator functions on  $[n]$ . If we want to quantify  $f(x) - f(y)$  in terms of  $\Delta(x, y)$ , for some  $f : \{0,1\}^n \rightarrow \mathbf{R}$ , then Fourier analysis becomes applicable, because  $f(x) - f(y) = (\sigma_x - \sigma_y)f(0)$ . Essentially all the problems and techniques considered in this book can be reduced to an analysis of the Hamming distance and its relation to some quantity under study.*

**Example.** A graph on  $n$  vertices can be identified with a Boolean string of length  $2^n$ , where each bit represents whether a particular edge is in  $G$ . The Hamming distance between two graphs is then the number of edges upon which the graphs differ. We may want to consider how quantities on graphs change under this Hamming distance, such as the shortest path between points, or the minimum cut, and Fourier analysis indicates a basic set of techniques to quantify this change.

**Example.** A subset of a set with  $n$  elements can be identified with a Boolean string of length  $n$ , which identifies which elements are present in the subset. The Hamming distance then measures the cardinality of the symmetric difference between two subsets, and this explains why Fourier analytic methods have applications in combinatorial problems in set theory.

**Example.** The truth values  $\top$  and  $\perp$  have an obvious relation to  $\{0,1\}$ , especially if you've used a modern programming language. The group operation on  $\{0,1\}$  then induces an operation on  $\top$  and  $\perp$ , which is the exclusive or operation

$$A \oplus B = \begin{cases} \top & A \text{ is } \top, \text{ or } B \text{ is } \top, \text{ but not both.} \\ \perp & \text{otherwise} \end{cases}$$

and where multiplication is conjunction. Given a property  $P$  of elements of some set  $X$ , this relation is used to form the indicator  $\mathbf{I}(P(x)) : X \rightarrow \{0,1\}$ , and then Fourier analytic methods can be used to analyze 'true Boolean functions'  $f : \{\top, \perp\}^n \rightarrow \{\top, \perp\}$ .

The most important correspondence with the additive structure of  $\{0,1\}$  in the general theory will be the multiplicative group  $\{-1,1\}$ , where we map 0 to 1, and 1 to -1. Initially, this is a very strange correspondence to pick, because we normally think of 1 as being the value of 'truth'. However, the correspondence does give an isomorphism between the two group structures, and fits with the general convention of performing harmonic analysis over the multiplicative group  $\mathbf{T}^n$  where possible. Rather than thinking of  $\{0,1\}$  as the canonical group upon which we take the correspondence, we shall find that  $\{-1,1\}$  will become a much more central figure in the analysis.

For instance, the characters on  $\{-1,1\}$  are notationally much more natural than the characters on  $\mathbf{F}_2^n$ . The characters on the group  $\mathbf{T}^n$  take the form of integer-valued monomials  $z_1^{m_1} \dots z_n^{m_n}$ . Because  $\{-1,1\}^n$  embeds as

a closed subgroup of  $\mathbf{T}^n$ , every character on  $\{-1, 1\}^n$  is the restriction of a character on  $\mathbf{T}^n$ , and can therefore be represented by a monomial. Since  $x_i^2 = 1$  for all  $x_i \in \{-1, 1\}$ , these monomials can be chosen with  $m_i \in \{0, 1\}$ . Abstract harmonic analysis tells us that there are exactly  $2^n$  characters on the group  $\{-1, 1\}$ , so these are all the identifications we can make, and therefore we conclude that the functions

$$x^S = \prod_{i \in S} x_i$$

form the set of all characters on  $\{-1, 1\}^n$ , for  $S \subset [n]$ . Using functional notation and switching domains, the characters on  $\mathbf{F}_2^n$  are

$$\chi_S(x) = (-1)^{\sum_{i \in S} x_i}$$

which isn't as natural a function.

The basic theory of discrete abelian harmonic analysis tells us that the characters  $x^S$  form an orthonormal basis for the Hilbert space of real-valued functions on  $\mathbf{F}_2^n$ , so every function  $f : \{-1, 1\}^n \rightarrow \mathbf{R}$  has a unique expansion

$$f(x) = \sum_{S \subset [n]} \hat{f}(S) x^S$$

where  $\hat{f}(S) \in \mathbf{R}$  is the *Walsh-Fourier coefficient* corresponding to  $S$ . It will also be natural to consider the Fourier transforms of functions on vector spaces  $\mathbf{F}_2^I$ , where  $I$  is an arbitrary finite index set. The expansion will then take the form

$$f(x) = \sum_{S \subset I} \hat{f}(S) x^S$$

We will occasionally use this notation when it is more natural to do so.

An easy way to find these coefficients, without any harmonic analysis, is to expand the any function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  in its conjunctive normal

form<sup>1</sup>, calculating

$$f(x) = \sum_y \mathbf{I}(x = y) f(y) = \sum_y \frac{1}{2^n} \left( \prod_{i=1}^n 1 + x_i y_i \right) f(y)$$

This formula gives us a *particular* expansion of the function, and we know it must be the *unique* expansion. This is essentially a binary equivalent to the technique used to find power series of holomorphic functions, combining well know expansions to form the expansion of a particular formula.

**Example.** The maximum function  $\max(x, y)$  on  $\{-1, 1\}^2$  has an expansion

$$\max(x, y) = \frac{1}{2} (1 + x + y - xy)$$

which corresponds to the conjunction function  $f(x, y) = x \wedge y$  on  $\{\top, \perp\}^2$ , or the minimum function on  $\mathbf{F}_2^2$ . To obtain this expansion, and a more general expansion for the maximum function  $\max_n$  on  $n$  variables note that we can write

$$\mathbf{I}(x = -1) = \frac{1}{2^n} \prod_{i=1}^n (1 - x_i) = \frac{1}{2^n} \sum_S (-1)^{|S|} x^S$$

so that

$$\begin{aligned} \max(x_1, \dots, x_n) &= \mathbf{I}(x \neq -1) - \mathbf{I}(x = -1) \\ &= 1 - 2\mathbf{I}(x = -1) \\ &= \left(1 - \frac{1}{2^{n-1}}\right) - \frac{1}{2^{n-1}} \sum_{S \neq \emptyset} (-1)^{|S|} x^S \end{aligned}$$

**Example.** Consider the minimum function  $\min(x) : \{-1, 1\}^n \rightarrow \{-1, 1\}$  on  $n$  bits, which corresponds to the disjunction operation on  $\{\top, \perp\}^n$ . Since we have

---

<sup>1</sup>If a property on  $\{-1, 1\}^n$  is identified with it's indicator function, then the indicator function of  $P \wedge Q$ , for any two properties  $P$  and  $Q$ , is obtained by multiplying together the corresponding indicator functions. Since the property  $x_i = y_i$  has indicator function  $\mathbf{I}(x_i = y_i) = (1/2)(1 + x_i y_i)$ , then we see that the expansion obtained for general functions  $f : \{-1, 1\}^n \rightarrow \{0, 1\}$  is exactly the one induced by the canonical conjunctive form expression  $P(x) = \bigvee_{P(y)} (y = x) = \bigvee_{P(y)} (\bigwedge_{i=1}^n y_i = x_i)$ , and we have extended this representation to general functions  $f : \{-1, 1\}^n \rightarrow \mathbf{R}$ .

the relationship  $\min(x) = -\max(-x)$ , we find

$$\begin{aligned} -\max(-x) &= -\left[ \left(1 - \frac{1}{2^{n-1}}\right) - \frac{1}{2^{n-1}} \sum_{S \neq \emptyset} (-1)^{|S|} (-x)^S \right] \\ &= \left( \frac{1}{2^{n-1}} - 1 \right) + \frac{1}{2^{n-1}} \sum_{S \neq \emptyset} x^S \end{aligned}$$

and this gives the expansion for  $\min(x)$ .

Note that the minimum and maximum functions on  $n$  variables are  $1/2^{n-1}$  close to constant functions in  $L^1$ , hence the Fourier transformations of the minimum and maximum functions are  $1/2^{n-1}$  close to the Fourier transforms of constant functions 1 and  $-1$  in  $L^\infty$ . Thus we see that the Fourier coefficients of  $\min$  and  $\max$  are bounded in absolute value by  $1/2^{n-1}$ , and the expected values of the functions are within a distance  $1/2^{n-1}$  from 1 and  $-1$ .

**Example.** In general, the indicator functions  $\mathbf{I}(x = a) : \{-1, 1\}^n \rightarrow \{0, 1\}$ , for some  $a \in \{-1, 1\}^n$  can be expressed as

$$\mathbf{I}(x = a) = \frac{1}{2^n} \prod_{i=1}^n (1 + x_i a_i) = \frac{1}{2^n} \sum_S a^S x^S$$

If we only have a subset  $X = \{i_1, \dots, i_k\}$  of indices we want to specify, then

$$\mathbf{I}(x_{i_1} = a_1, \dots, x_{i_k} = a_k) = \frac{1}{2^k} \sum_{S \subset X} a^S x^S$$

Thus the cardinality of sets  $S$  such that  $\hat{f}(S) \neq 0$  is bounded by  $k$ .

**Example.** Consider  $n$  Bernoulli distributions over  $\{-1, 1\}$  with means  $\mu_1, \dots, \mu_n$ , and consider the corresponding product distribution on  $\{-1, 1\}^n$ , with density function  $f$ . If  $X$  is a random variable with this distribution, then

$$\mathbf{P}(X_i = 1) - \mathbf{P}(X_i = -1) = 2\mathbf{P}(X_i = 1) - 1 = \mu_i$$

so  $\mathbf{P}(X_i = 1) = \frac{1}{2}(\mu_i + 1) = P_i \in [0, 1]$ . We then calculate

$$\begin{aligned}
f(x_1, \dots, x_n) &= \prod_{i=1}^n [P_i \mathbf{I}(x_i = 1) + (1 - P_i) \mathbf{I}(x_i = -1)] \\
&= \sum_a \left( \prod_{a_i=1} P_i \right) \left( \prod_{a_i=-1} (1 - P_i) \right) \mathbf{I}(x = a) \\
&= \frac{1}{2^n} \sum_S \sum_a a^S \left( \prod_{a_i=1} P_i \right) \left( \prod_{a_i=-1} (1 - P_i) \right) x^S \\
&= \frac{1}{2^n} \sum_S \left( \prod_{i \in S} (2P_i - 1) \right) x^S \\
&= \frac{1}{2^n} \sum_S \mu^S x^S
\end{aligned}$$

where we obtain the second last equation by repeatedly factoring out the coordinates  $a_i$  for  $a_i = 1$  and  $a_i = -1$ .

**Example.** Define the ‘inner product’ function  $f$  on  $\mathbf{F}_2^n \times \mathbf{F}_2^n$  by

$$f(x, y) = (-1)^{\langle x, y \rangle} = \prod_{i=1}^n (-1)^{x_i y_i}$$

which is a bilinear map from  $\mathbf{F}_2$  to  $\{-1, 1\}$ . This corresponds to the function  $g$  on  $\{-1, 1\}^n \times \{-1, 1\}^n$  defined by

$$g(x, y) = \prod_{i=1}^n (-1)^{\mathbf{I}(x_i = y_i = -1)}$$

For  $n = 1$ , we have

$$g(x, y) = \begin{cases} -1 & x = y = -1 \\ 1 & \text{otherwise} \end{cases}$$

Hence  $g(x, y)$  is just the max function on  $\{-1, 1\}^2$ , and we find

$$g(x, y) = (-1)^{\mathbf{I}(x=y=-1)} = \frac{1}{2}(1 + x + y - xy)$$



In general, any subset of indices on  $\{1, 2\} \times [n]$  (this is the index set for  $\{-1, 1\}^n \times \{-1, 1\}^n$ ) can be decomposed as  $S = \{1\} \times S_1 \cup \{2\} \times S_2$ , where  $S_1, S_2 \subset [n]$ . Let  $\alpha(S)$  be the cardinality of  $S_1 \cap S_2$ . Then we have

$$\begin{aligned}
g(x, y) &= \frac{1}{2^n} \prod_{i=1}^n (1 + x_i + y_i - x_i y_i) \\
&= \frac{1}{2^n} \sum_{S \subset [n]} (-1)^{|S|} (xy)^S (1 + x + y)^{S^c} \\
&= \frac{1}{2^n} \sum_{S \subset [n]} (-1)^{|S|} \sum_{T \subset S^c} (xy)^S (x + y)^T \\
&= \frac{1}{2^n} \sum_{S \subset \{1, 2\} \times [n]} (-1)^{\alpha(S)} (x, y)^S
\end{aligned}$$

Note that

$$g(x, x) = \frac{1}{2^n} \sum_S (-1)^{\alpha(S)} x^{S_1 \Delta S_2}$$

The number of  $S_1 \Delta S_2 = [n]$  is exactly the number of partitions of  $[n]$ , which are  $2^n$ , and these partitions must be disjoint, hence the corresponding  $\alpha(S)$  is always equal to 0, hence  $x^{[n]}$  has a coefficient of 1. Conversely, if  $T$  is a proper subset of  $[n]$ , not containing some element  $x$ , then the set of  $S_1, S_2$  can be divided in half into the family such that  $x \in S_1$  and  $x \in S_2$  and the family such that  $x \notin S_1 \cup S_2$ . The coefficient  $(-1)^{\alpha(S)}$  on one family is the opposite of the coefficient of the other family, so when we sum over all possible coefficients, we find that the coefficient of  $g(x, x)$  corresponding to  $x^T$  is zero. This makes sense, since  $g(x, x) = x^{[n]}$ .

**Example.** The equality function  $Eq$  returns 1 only if all  $x_i$  are equal. Thus

$$\begin{aligned}
Eq(x) &= \mathbf{I}(x_1 = x_2 = \dots = x_n = 1) + \mathbf{I}(x_1 = x_2 = \dots = x_n = -1) \\
&= \frac{1}{2^n} \sum_S (1 + (-1)^{|S|}) x^S = \frac{1}{2^{n-1}} \sum_{|S| \text{ even}} x^S
\end{aligned}$$

The not all equal function  $Neq$  returns 1 only if all  $x_i$  not all equal. Thus

$$Neq(x) = 1 - Eq(x) = \left(1 - \frac{1}{2^{n-1}}\right) - \sum_{\substack{|S| \text{ even} \\ S \neq \emptyset}} x^S$$

**Example.** The sortedness function  $\text{Sort} : \{-1, 1\}^4 \rightarrow \{-1, 1\}$  returns -1 if the bits in the input are monotonically increasing or monotonically decreasing. Since the function is even, all odd Fourier coefficients vanish. It will also help that  $\text{Sort}$  is invariant under the coordinate permutation (14)(23), hence the Fourier coefficients obtained by swapping these coordinates are equal. To calculate the coefficients, it will be helpful to switch to calculating the coefficients of  $f = (1 - \text{Sort})/2$ , which is now  $\{0, 1\}$  valued.  $f(x) = 1$  if and only if  $x$  is one of the following 8 bit strings

$$\begin{aligned} &(-1, -1, -1, -1), (-1, -1, -1, +1), (-1, -1, +1, +1), (-1, +1, +1, +1) \\ &(+1, +1, +1, +1), (+1, +1, +1, -1), (+1, +1, -1, -1), (+1, -1, -1, -1) \end{aligned}$$

Hence the expected value of  $f$  is  $1/2$ . We also calculate that

$$\begin{aligned} \hat{f}(\{1, 2, 3, 4\}) &= 0 & \hat{f}(\{1, 2\}) &= \hat{f}(\{3, 4\}) = 1/4 \\ \hat{f}(\{1, 3\}) &= \hat{f}(\{2, 4\}) = 0 & \hat{f}(\{1, 4\}) &= -1/4 \\ \hat{f}(\{2, 3\}) &= 1/4 \end{aligned}$$

Thus  $f(x) = (1/4)(2 + x_1x_2 + x_3x_4 + x_2x_3 - x_1x_4)$ , and so

$$\text{Sort}(x) = 1 - 2f(x) = (1/2)(x_1x_4 - x_1x_2 - x_2x_3 - x_3x_4)$$

The  $x_1x_4$  term determines whether  $x$  should be the constant 1 or  $-1$  term, and the rest offset the problem by pairwise comparisons.

**Example.** The hemi-icosohedron function  $HI : \{-1, 1\}^6 \rightarrow \{-1, 1\}$  takes a set of labels for the vertices of the hemi-icosohedron graph below, and returns the number of faces labelled  $(1, 1, 1)$ , minus the number of faces labelled  $(-1, -1, -1)$ , modulo 3 (TO DO: FIND PICTURE OF HEMIICOSOHEDRON, OR DRAW ONE)-. The graph is homogenous, in the sense that there is a graph isomorphism mapping a vertex to any other vertex. In fact, one can obtain all isomorphisms by choosing which face one face will be mapped to, and specifying where two of the vertices on that face will go in the new face.

For any  $x$ , if we consider  $-x$ , then any face labelled  $(1, 1, 1)$  becomes a face labelled  $(-1, -1, -1)$ , and vice versa, so that  $HI(-x) = -HI(x)$ . Thus  $HI$  is an odd function, and all even Fourier coefficients vanish. It will help to calculate  $HI$  explicitly. Note that  $HI(-1) = -1$ , because all 10 faces are labelled  $(-1, -1, -1)$ , and  $H(1) = 1$ . If  $x$  has a single 1, then 5 of the faces are labelled

$(-1, -1, -1)$ , and none are labelled  $(1, 1, 1)$ , so  $HI(x) = 1$ . If  $x$  has exactly 2 ones, then we see that only two faces are labelled  $(-1, -1, -1)$ , and none are labelled  $(1, 1, 1)$ , so  $HI(x) = 1$ . If  $x$  has three ones arranged in a triangle on the hemi-icosohedron, then the hemi-icosohedron has one face labelled  $(1, 1, 1)$ , and none labelled  $(-1, -1, -1)$ , and hence  $HI(x) = 1$ . If  $x$  does not have three ones arranged in a triangle, then we find it must have three negative ones arranged in a triangle, hence  $HI(x) = -1$ . Since  $HI$  is odd, this calculates  $HI$  explicitly for all values of  $x$ .

Since there is an isomorphism mapping any vertex to any other vertex, the degree one Fourier coefficients of  $HI$  are all equal, and we might as well calculate the coefficient corresponding to  $x_1$ . Now

$$\widehat{HI}(1) = \mathbf{E}[HI(X)X_1] = \frac{\mathbf{E}[HI(X)|X_1 = 1] - \mathbf{E}[HI(X)|X_1 = -1]}{2}$$

Since  $HI$  is odd,

$$\mathbf{E}[HI(X)|X_1 = -1] = -\mathbf{E}[HI(-X)|X_1 = -1] = -\mathbf{E}[HI(X)|X_1 = 1]$$

hence  $\widehat{HI}(1) = \mathbf{E}[HI(X)|X_1 = 1]$ . If we let  $Y$  be the random variable denoting the number of  $x_i = 1$ , for  $i \neq 1$ , then

$$\begin{aligned} \mathbf{E}[HI(X)|X_1 = 1] &= (1/32) \sum_{k=0}^5 \binom{5}{k} \mathbf{E}[HI(X)|X_1 = 1, Y = k] \\ &= (1/32) \left( \binom{5}{0} + \binom{5}{1} - \binom{5}{3} - \binom{5}{4} + \binom{5}{5} \right) \\ &= (1/32)(1 + 5 - 10 - 5 + 1) = -1/4 \end{aligned}$$

since  $\mathbf{E}[HI(X)|X_1 = 1, Y = 2] = (1/10)(5 - 5) = 0$ . To simplify the calculation of coefficients of degree, note that we have graph isomorphisms mapping any triangle to any other triangle, hence the Fourier coefficients of degree 3 corresponding to a triangle are all equal. If  $Y$  is the number of  $x_i = 1$ , for  $i \neq 1, 2, 3$ , then by applying the same technique as when we were calculating

$\mathbf{E}[HI(X)|X_1 = 1]$ , we find

$$\mathbf{E}[HI(X)|X_1 = +1, X_2 = +1, X_3 = +1] = (1/8)(1 - 3 - 3 + 1) = -1/2$$

$$\mathbf{E}[HI(X)|X_1 = +1, X_2 = +1, X_3 = -1] = (1/8)(1 + 1 - 2 - 3 - 1) = -1/2$$

$$\begin{aligned} \mathbf{E}[HI(X)|X_1 = +1, X_2 = -1, X_3 = -1] &= -\mathbf{E}[HI(X)|X_1 = +1, X_2 = +1, X_3 = -1] \\ &= 1/2 \end{aligned}$$

$$\begin{aligned} \mathbf{E}[HI(X)|X_1 = -1, X_2 = -1, X_3 = -1] &= -\mathbf{E}[HI(X)|X_1 = +1, X_2 = +1, X_3 = +1] \\ &= 1/2 \end{aligned}$$

We can then interpolate these results to find

$$\hat{f}(\{1, 2, 3\}) = (-1/2 + 3/2 + 3/2 - 1/2)(1/8) = 1/4$$

Now the sum of the squares of the Fourier coefficients we have calculated are  $10(1/4)^2 + 6(1/4)^2 = 1$ , so that all other Fourier coefficients are zero. Thus

$$f(x) = 1/4(e_\Delta - e_1)$$

When  $e_1$  is the first symmetric polynomial (the sum of all monomials of degree one), and  $e_\Delta$  is the sum of all monomials which form triangles.

## 1.2 Fourier Analysis and Probability Theory

Many of the interesting results of Boolean Fourier analysis occur in the context of statistical problems over discrete domains. Thus it is of interest to interpret the main tools and results of probability theory in the context of Boolean analysis. Unless otherwise stated, the distributions of all results will be uniform. This means the probability theory rephrases combinatorial results, but it is still a useful source of intuition and elegant notation.

First, note that we can define an inner product on Boolean functions by letting

$$\langle f, g \rangle = \frac{1}{2^n} \sum f(x)g(x) = \mathbf{E}[f(X)g(X)]$$

The characters on  $\{-1, 1\}^n$  form an orthonormal basis to this inner product, and it therefore follows that for any Boolean function  $f$ ,

$$\hat{f}(S) = \mathbf{E}[f(X)X^S]$$

hence the Fourier coefficients of  $f$  correspond to the average value of  $f$  relative to the parity of  $S$ . For instance, a Boolean-valued function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  is *unbiased*, that is, it takes values 1 and -1 with equal probability, if and only if  $\hat{f}(\emptyset) = 0$ .

The standard Hilbert space results give rise to probabilistic interpretations. Parseval's equality means

$$\mathbf{E}[f(X)^2] = \sum \hat{f}(S)^2$$

which implies that

$$\mathbf{V}[f(X)] = \mathbf{E}[f(X)^2] - \mathbf{E}[f(X)]^2 = \sum_{S \neq \emptyset} \hat{f}(S)^2$$

$$\text{Cov}(f(X), g(X)) = \sum_{S \neq \emptyset} \hat{f}(S) \hat{g}(S)$$

Hence expectations, variances, and covariances are directly related to the Fourier coefficients of the Boolean function in question.

If  $f$  and  $g$  are *Boolean-valued maps*, then we can write

$$\mathbf{E}[f(X)g(X)] = \mathbf{P}(f(X) = g(X)) - \mathbf{P}(f(X) \neq g(X)) = 2\mathbf{P}(f(X) = g(X)) - 1$$

Note that  $\mathbf{P}(f(X) = g(X))$  differs from the Hamming distance of  $f$  and  $g$  by a constant. We define this value to be the **relative Hamming distance**  $d(f, g)$ . Since  $f^2 = g^2 = 1$ ,  $\|f\|_2^2 = 1$ , and this implies that

$$\begin{aligned} \mathbf{V}(f) &= \mathbf{E}[f(X)^2] - \mathbf{E}[f(X)]^2 \\ &= 1 - (\mathbf{P}(f(X) = 1) - \mathbf{P}(f(X) = -1))^2 \\ &= 4d(f, 1)d(f, -1) \end{aligned}$$

Thus the variance of Boolean valued functions is very closely related to the degree to which the function is constant.

**Lemma 1.1.** *If  $\varepsilon = \min[d(f, 1), d(f, -1)]$ , then  $2\varepsilon \leq \mathbf{V}(f) \leq 4\varepsilon$ .*

*Proof.* We are effectively proving that for any  $x \in [0, 1]$ ,

$$2 \min(x, 1 - x) \leq 4x(1 - x) \leq 4 \min(x, 1 - x)$$

Since  $4x(1 - x) = 4 \max(x, 1 - x) \min(x, 1 - x)$ , we divide by  $4 \min(x, 1 - x)$  and restate the inequality as  $1/2 \leq \max(x, 1 - x) \leq 1$ , which is obvious.  $\square$

This shows that if we have a sequence of functions  $f_n : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , then  $\varepsilon_n \sim x_n$  holds if and only if  $\mathbf{V}(f) \sim x_n$ . The minimum hamming distance to a constant value is asymptotically the same as the variation of the function. In general domains we intuitively view  $\mathbf{V}(f)$  as a measure of how constant the function  $f$  is, but for functions  $\{-1, 1\}$  we have a precise, quantitative estimate.

**Example.** Consider a function  $f$  chosen uniformly randomly from the set of all Boolean functions on  $\{-1, 1\}^n$ . That is, for any function  $g : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , the probability distribution gives  $\mathbf{P}(f = g) = 1/2^{2^n}$ . Since each Boolean function has a Fourier expansion,  $f$  has a Boolean expansion

$$f(x) = \sum \hat{f}(S) x^S$$

where each  $\hat{f}(S)$  is now a real-valued random variable. Now because we have a bijection amongst the set of all Boolean functions, mapping  $g$  to  $-g$ , we find

$$\begin{aligned} \mathbf{E}[\hat{f}(S)] &= \frac{1}{2^{2^n}} \sum_g \hat{g}(S) = \frac{1}{2^{2^n}} \sum (\widehat{-g})(S) \\ &= -\frac{1}{2^{2^n}} \sum \hat{g}(S) = -\mathbf{E}[\hat{f}(S)] \end{aligned}$$

Hence  $\mathbf{E}[\hat{f}(S)] = 0$ . What's more, given that  $X \neq Y$ , the probability that  $f(X) = f(Y)$  is independent of  $X$  and  $Y$ , once we know  $X \neq Y$ , hence

$$\begin{aligned} \mathbf{V}[\hat{f}(S)] &= \mathbf{E}[\hat{f}(S)^2] = \mathbf{E} \left[ \mathbf{E} \left( f(X) X^S \right)^2 \right] \\ &= \mathbf{E}[f(X)f(Y)(XY)^S] \\ &= \mathbf{P}(X = Y) + \mathbf{P}(X \neq Y) \mathbf{E}[f(X)f(Y)(XY)^S | X \neq Y] \\ &= \frac{1}{2^n} + \left(1 - \frac{1}{2^n}\right) (2\mathbf{P}(f(X) = f(Y) | X \neq Y) - 1) \mathbf{E}((XY)^S | X \neq Y) \end{aligned}$$

For any particular  $x \neq y$ ,  $\mathbf{P}[f(x) = f(y)] = 1/2$ , because  $f$  is chosen uniformly from all functions, and for any function  $g$ , we have the function  $g'$ , such that  $g'(z) = g(z)$  except when  $z = x$ , where  $g'(z) = -g(x)$ , and the association is a bijection. This implies that  $\mathbf{P}(f(X) = f(Y) | X \neq Y) = 1/2$ , hence the variation satisfies  $\mathbf{V}[\hat{f}(S)] = 1/2^n$ . Thus functions on average have no Fourier coefficient on each  $S$ , and most functions have very small Fourier coefficient.

Since a function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  always has square-norm 1, because  $f^2 = 1$ , Parseval's theorem implies that  $\sum \hat{f}(S)^2 = 1$  if we sum over all the Fourier coefficients, so that a boolean-valued function on  $n$  variables gives rise to a probability distribution over  $[n]$ . We call this the spectral sample of  $f$ , and denote the distribution by  $\mathcal{S}_f$ .

Often times, the Fourier coefficients of sets over some particular cardinality are more than enough to determine some result. We define the weight of a function  $f : \{-1, 1\}^n \rightarrow \mathbf{R}$  of degree  $k$  to be

$$\mathbf{W}^k(f) = \sum_{|S|=k} \hat{f}^2(S)$$

If  $f$  is boolean-valued, then we can also define  $\mathbf{W}^k(f) = \mathbf{P}(|S| = k)$  where  $S$  is a set randomly drawn from the spectral sample of  $f$ .  $\mathbf{W}^k(f)$  is also the square norm of the function

$$(P^k f)(x) = \sum_{|S|=k} \hat{f}(S) x^S$$

which can be seen as a certain truncation estimate of  $f$ .

### 1.3 Testing Linearity

A function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  is linear if  $f(xy) = f(x)f(y)$  for all  $x, y \in \{-1, 1\}^n$ , or equivalently, if there is  $S \subset [n]$  such that  $f(x) = x^S$ . Given a general function  $f$ , there are effectively only two ways to explicitly check that the function is linear, we either check that  $f(xy) = f(x)f(y)$  for all choices of the arguments  $x$  and  $y$ , or check that  $f(x) = x^S$  holds for some choice of  $S$ , over all choices of  $x$ . Even assuming that  $f$  can be evaluated in constant time, both methods take exponential time in  $n$  to compute. This is guaranteed, because the minimal description length of  $f$  is always exponential in  $n$ , and if an algorithm determining linearity does not check the entire description of a linear function  $f$ , we can modify  $f$  to a non-linear function on inputs that the linearity tester doesn't look at, at the algorithm will not be able to distinguish between these two inputs. The problem of testing linearity is a scenario in a family of problems in the field of **property testing**, which attempts to design efficient algorithms to

determine whether a particular boolean-valued function satisfies a certain property.

We might not be able to come up with a polynomial time algorithm to verify linearity, but we can make headway by considering the possibility of coming up with a randomized algorithm which can verify linearity with high probability. The likely solution to the problem, given some function  $f$ , would to perform the linearity test for  $f(XY) = f(X)f(Y)$  for a certain set of randomly chosen inputs  $X$  and  $Y$ . If  $f$  is non-linear, then  $f(XY) \neq f(X)f(Y)$  with positive probability, and if we find this particular input we can guarantee the function is non-linear. If  $f$  is linear, the test always passes. We shall find that the probability that the test fails directly relates to how similar a function is to a linear function.

The simplest version of linearity testing runs using one random query as a test – it just takes one pair of inputs  $(X, Y)$ , and tests the linearity of this function against these inputs. This is known as the Blum-Luby-Rosenfeld algorithm, or BLR for short. It turns out that the success of this method is directly related to how similar a function is to a linear character. There are two candidates to define what it means to be ‘approximately linear’.

- $f(xy) = f(x)f(y)$  for a large majority of inputs  $x$  and  $y$ . Essentially, this means exactly that  $f(XY) = f(X)f(Y)$  with high probability, so the BLR test is accepted with high probability on input  $f$ .
- There is a linear functional  $g$  such that  $f(x) = g(x)$  for a large number of inputs  $x$ . This means exactly that  $d(f, g)$  is small.

The two notions are certainly related, but a direct relation seems difficult to quantify. One way is easy to bound. If  $g$  is linear, and  $d(f, g) = \epsilon$ , then a standard union bound inequality guarantees that

$$\begin{aligned} \mathbf{P}[f(X)f(Y) = f(XY)] &\geq \mathbf{P}[f(X) = g(X), f(Y) = g(Y), f(XY) = g(XY)] \\ &\geq 1 - 3\epsilon \end{aligned}$$

and we shall find that even tighter bounds hold. But just because  $f(xy) = f(x)f(y)$  for a large number of  $x, y$  doesn’t seem to imply that  $f$  is close to any *particular* linear function. For a subset  $\mathcal{X}$  of Boolean-valued functions, we shall let  $d(f, \mathcal{X}) = \inf_{g \in \mathcal{X}} d(f, g)$ . If  $\mathcal{C}$  is the set of characters, then we can state our problem as analyzing the relation between how the BLR test interacts with  $f$ , and the value of  $d(f, \mathcal{C})$ .



Our analysis of the BLR test essentially shows that there is a very close relation between these two properties for the case of Boolean functions. Since we are applying Hamming distances in measuring how linear a function is, we can guess that the Harmonic analysis of boolean functions will come in handy.

**Theorem 1.2.** *For any function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , if  $d(f, \mathcal{C}) \geq \varepsilon$ , then the BLR algorithm rejects  $f$  with probability greater than  $\varepsilon$ .*

*Proof.* Note that

$$\mathbf{I}[f(XY) \neq f(X)f(Y)] = \frac{1 - f(X)f(Y)f(XY)}{2}$$

Hence

$$\begin{aligned} \mathbf{P}[\text{BLR rejects } f] &= \mathbf{E} \left[ \frac{1 - f(X)f(Y)f(XY)}{2} \right] \\ &= \frac{1}{2} - \frac{1}{2} \mathbf{E}_X[f(X) \mathbf{E}_Y[f(Y)f(XY)]] \\ &= \frac{1}{2} - \frac{1}{2} \mathbf{E}_X[f(X)(f * f)(X)] \\ &= \frac{1}{2} - \frac{1}{2} \sum \hat{f}(S)^3 \\ &\geq \frac{1}{2} - \frac{1}{2} \max \hat{f}(T) \sum \hat{f}(S)^2 \\ &= \frac{1}{2} - \frac{1}{2} \max \hat{f}(T) \end{aligned}$$

Now for each  $S$ ,  $\hat{f}(S) = 1 - 2d(f, x^S)$ , so if  $d(f, x^S) \geq \varepsilon$  for all  $S$ , we find  $\hat{f}(S) \leq 1 - 2\varepsilon$ , hence the probability that the BLR test rejects  $f$  is greater than

$$\mathbf{P}[\text{BLR rejects } f] \geq \frac{1}{2} - \frac{1}{2}(1 - 2\varepsilon) = \varepsilon$$

and this completes the proof. The converse of this result is that if the BLR test accepts  $f$  with probability  $1 - \varepsilon$ , then  $f$  is  $\varepsilon$  close to being linear, and so we have shown that  $f(XY) = f(X)f(Y)$  holds with high probability if and only if  $f$  is  $\varepsilon$  close to being linear.  $\square$

Note that this algorithm, with only three evaluations of the function  $f$ , can determine with high accuracy that  $f$  is not linear, if the function is far

away from being non-linear to begin with, or, if we are wrong, then  $f$  is very similar to a linear function in the first case. Yet given  $f$ , we cannot use this algorithm to determine the linear function  $x^S$  which  $f$  is similar to. Of course, for most  $x$ ,  $f(x) = x^S$ , but we do not know for which  $x$  this occurs. However, we cannot use this estimate to obtain accurate estimates in expectation for a fixed  $x$ , in the sense that there is no measure of the likelihood of the estimate being equal to the actual value. The problem is that  $x$  is a fixed quantity, rather than varying over many values of the function, and is therefore our test is easy to break. Note, however, that  $y^S = x^S(xy)^S$ , and if we let  $x$  be a random quantity, then  $(xy)^S$  and  $x^S$  will likely be equal to  $f(xy)$  and  $f(y)$  with a probability that is feasible to determine.

**Theorem 1.3.** *If  $d(f, x^S) = \varepsilon$ , then for any  $x$ ,*

$$\mathbf{P}[f(xY)f(Y) = x^S] \geq 1 - 2\varepsilon$$

*Proof.* We just apply a union bound to conclude

$$\mathbf{P}(f(xY)f(Y) = x^S) \geq \mathbf{P}(f(xY) = x^S Y^S, f(Y) = Y^S) \geq 1 - 2\varepsilon$$

and in this circumstance, we find  $f(yX)f(X) = y^S$ .  $\square$

Thus linear functions are *locally correctable*, in that we can correct small modifications to linear functions with high probability. This turns out to have vast repercussions in the theory of property testing and complexity theory, which we will discuss later. One reason why these methods work is that the linear functions on  $\mathbf{F}_2^n$  form a very discrete set. For  $S \neq T$ ,

$$d(x^S, x^T) = \mathbf{P}(x^S \neq x^T) = \mathbf{P}(x^{S \Delta T} = -1) = 1/2$$

We find characters are separated by a dimension-independant quantity. If  $d(f, x^T) \leq \varepsilon$  for some  $T$ , then  $d(f, x^S) \geq 1/2 - \varepsilon$  for  $S \neq T$ , and therefore it is difficult to splice together two linear functions so that  $f(XY) = f(X)f(Y)$  holds with high probability without  $f$  being close to a particular linear function.

The inequality in our analysis of the BLR test is tight, provided that all  $\hat{f}(S)$  are reasonably equal to one another. Analyzing the gap in the inequality, we find

$$\begin{aligned} \max \hat{f}(T) \sum \hat{f}(S)^2 - \sum \hat{f}(S)^3 &= \sum \hat{f}(S)^2 [\max \hat{f}(T) - \hat{f}(S)] \\ &\leq \max \hat{f}(T) - \min \hat{f}(S) \end{aligned}$$

If all Fourier coefficients are equal, then the inequality is an equality. Though this isn't strictly the spectra of a *Boolean-valued* function (it's the spectra of the indicator function  $\mathbf{I}(x = 1)$ ), we can find Boolean functions  $f$  whose Fourier coefficients are arbitrarily close to 0. For instance, the inner product function we considered above shows

$\hat{f}(S) = \hat{f}(T)$  for all  $S$  and  $T$ , then the inequality is an equality. Though this isn't the spectra of a *Boolean-valued* function (it is the spectra of the indicator function  $\mathbf{I}(x = 1)$ ), we can find Boolean functions  $f$  whose Fourier coefficients are arbitrarily close. Thus we have found the best dimension independant constant that holds over all functions  $f$ . However, if one of the  $\hat{f}(S)$  is much larger than the others, so  $f$  is very close to  $x^S$ , then we can obtain a better bound.

**Theorem 1.4.** *If  $d(f, x^S) = \varepsilon$ , then the BLR test rejects  $f$  with probability at least  $3\varepsilon - 10\varepsilon^2 + 8\varepsilon^3$ .*

*Proof.* Note that

$$\hat{f}(S) = \mathbf{E}[f(X)X^S] = 1 - 2d(f, x^S) = 1 - 2\varepsilon$$

and also for  $T \neq S$ ,

$$\hat{f}(T) = 1 - 2d(f, x^T) \leq 1 - 2(d(f, x^S) - d(x^T, x^S)) = 2\varepsilon$$

Hence

$$\begin{aligned} \mathbf{P}(\text{BLR rejects } f) &= \mathbf{E} \left( \frac{1 - f(X)f(Y)f(XY)}{2} \right) \\ &= \frac{1}{2} - \frac{1}{2} \sum_{T \neq S} \hat{f}(T)^3 - \frac{1}{2} \hat{f}(S)^3 \\ &\geq \frac{1}{2} - \varepsilon \sum_{T \neq S} \hat{f}(T)^2 - \frac{1}{2} \hat{f}(S)^3 \\ &= \frac{1}{2} - \varepsilon(1 - \hat{f}(S)^2) - \frac{1}{2} \hat{f}(S)^3 \\ &= \frac{1}{2} - \varepsilon(1 - (1 - 2\varepsilon)^2) - \frac{1}{2}(1 - 2\varepsilon)^3 \\ &= 3\varepsilon - 10\varepsilon^2 + 8\varepsilon^3 \end{aligned}$$

□

The converse of this statement is that if the BLR test accepts  $f$  with probability greater than  $1 - \varepsilon + O(\varepsilon^2)$ , then  $f$  is  $\varepsilon/3$  close to being linear. Thus for small  $\varepsilon$  the bound on similarity to a linear function is tighter than we initially calculated. This makes sense because, when  $f$  is very close to a linear function, it seems intuitive that it is difficult for  $f(xy) = f(x)f(y)$  *not* to hold.

This is the best upper bound we can obtain, up to a first order. For each  $n$ , consider the function  $g_n : \mathbf{F}_2^n \rightarrow \mathbf{F}_2$ , with  $g_n(x) = \mathbf{I}(x = 1)$ . Let us count all instances when  $g_n$  is rejected by the BLR test. There are three possibilities where  $g_n(x+y) \neq g_n(x) + g_n(y)$ :

- If  $x = 1$  and  $y \neq 1$ .
- If  $x \neq 1$  and  $y = 1$ .
- If  $x \neq 1$  and  $y \neq 1$ , and  $x + y = 1$ .

There are  $2^n - 1$  instances of the first and second rejection, and  $2^n - 2$  instances of the third. Thus

$$\mathbf{P}(\text{BLR rejects } g_n) = \frac{2(2^n - 1) + (2^n - 2)}{4^n} = \frac{3}{2^n} - \frac{4}{4^n}$$

and  $g_n$  is  $1/2^n$  close to 0. If there was a lower bound of the form  $C\varepsilon - O(\varepsilon^2)$ , for  $C > 3$ , and if we write  $1/2^n = \varepsilon$ , then for big enough  $n$  we find  $3\varepsilon - 4\varepsilon^2 \geq (3 + \delta)\varepsilon$ , hence  $-4\varepsilon \geq \delta$ , which is impossible since we can let  $\varepsilon$  tend to zero.

## 1.4 The Walsh-Fourier Transform

Historically, the study of the Fourier transform on  $\mathbf{F}_2^n$  emerged from a novel analysis of the Fourier expansions of  $L^2[0,1]$ . In this section we consider this development for completeness. Consider the product group  $\mathbf{F}_2^\infty = \prod \mathbf{F}_2$ , which is a compact abelian group. We have the standard projections  $i_n : \mathbf{F}_2^n \rightarrow \mathbf{F}_2^\infty$ , defined by  $i_n(x_1, \dots, x_n) = (x_1, \dots, x_n, 0, 0, \dots)$ . In addition, we have the projections  $\pi_n(x_1, x_2, \dots) = (x_1, \dots, x_n)$ . Given a character  $f$  on  $\mathbf{F}_2^\infty$ , each  $f \circ i_n$  is a character on  $\mathbf{F}_2^n$ , and can thus be written  $\chi_{S_n}$  for some  $S_n \subset [n]$ . The sets  $S_n$  defined must be increasing in  $n$ , and for  $n \leq m$ , satisfy  $S_m \cap [n] = S_n$ . For any  $x = (x_1, x_2, \dots)$  in  $\mathbf{F}_2^\infty$ , the sequence  $\pi_n(x)$  converges to  $x$ , and hence

$$f(x) = \lim \pi_n(x) = \lim x^{S_n}$$

If  $S = \lim S_n$  is not a bounded subset of the integers, then the limit above can be allowed to oscillate infinitely between  $-1$  and  $1$ , hence  $f$  cannot be a *continuous* character on  $\mathbf{F}_2^\infty$ . We therefore find that characters on  $\mathbf{F}_2^\infty$  correspond to finite subset of positive integers, and topologically is essentially the inductive limit of the character groups  $(\mathbf{F}_2^n)^*$ .

An important link between probability theory and real analysis is that the binary expansion of real numbers on  $[0, 1]$ ,

$$x = \sum_{m=1}^{\infty} \frac{x_m}{2^m}$$

can be seen as defining a measure preserving map between  $\mathbf{F}_2^\infty$  and  $[0, 1]$ , where

$$(x_1, x_2, \dots) \mapsto \sum_{m=1}^{\infty} \frac{x_m}{2^m}$$

which is injective almost everywhere. Thus we can find a uniform distribution over  $[0, 1]$  by taking the binary expansion obtained by flipping a fair coin infinitely many times. Important to our work is that the harmonic analysis of  $\mathbf{F}_2^\infty$  gives rise to an expansion theory on  $[0, 1]$ . In particular, we find that the characters  $\chi_S$ , which form an orthonormal basis for  $L^2(\mathbf{F}_2^\infty)$  correspond to an orthonormal basis of functions on  $L^2[0, 1]$  of the form

$$w^S(x) = \chi_S(x_1, x_2, \dots) = \prod_{i \in S} (-1)^{x_i} = \prod_{i \in S} r_i(x)$$

where  $r_i(x) = (-1)^{x_i}$  is the  $i$ 'th Rademacher function. Because integers have unique binary expansions, we may reindex  $w^S$  as  $w_n$  for a unique non-negative integer  $n$ , as was done classically, and this family of functions are called the Walsh functions. Thus every function has a unique expansion

$$f(x) = \sum_{n=0}^{\infty} a_n w_n(x) = \sum_{\substack{S \subseteq [n] \\ \#S < \infty}} \hat{f}(S) w^S(x)$$

and this is essentially where the beginnings of the study of Boolean expansions began, by Walsh in the 1920s. These expansions have very good  $L^p$  truncations, but they are less used in modern mathematics because the functions  $w_n$  are not smooth, and as such are not useful in the study of partial differential equations.

For the purposes of discrete Boolean analysis, we shall be interested in this system in order to come up with a computationally efficient way of computing the Walsh-Fourier expansion of a boolean valued function  $f : \mathbf{F}_2^n \rightarrow \mathbf{R}$ . For an integer  $k$ , define  $k_n$  to be the coefficient in the binary expansion

$$k = \sum k_n 2^n$$

First, note that there is a basis  $\{X_k\}$  of functions from  $\mathbf{F}_2^n \rightarrow \mathbf{R}$  defined by

$$X_k = \mathbf{I}(x_1 = k_1, \dots, x_n = k_n)$$

and a basis  $\{Y_k\}$  of elements of functions from  $\mathbf{F}_2^*$  to  $\mathbf{R}$  defined by

$$Y_k = \chi_{\{i:k_i=1\}}$$

Since the Walsh transform  $\mathcal{W}$  is effectively a map from  $\mathbf{R}^{\mathbf{F}_2^n}$  to  $\mathbf{R}^{(\mathbf{F}_2^n)^*}$ , these bases induce a matrix representation  $W$  of the transform. We have

$$\mathcal{W}(X_l) = \frac{1}{2^n} \sum_S (-1)^{\sum_{m \in S} l_m} \chi_S = \frac{1}{2^n} \sum_k (-1)^{\sum l_i k_i} Y_k$$

Hence if we define the matrices

$$H^1 = (1)$$

$$H^{n+1} = \begin{pmatrix} H^n & H^n \\ H^n & -H^n \end{pmatrix}$$

Then  $W = H_n/2^n$ , because, as can be proved by induction,  $H_{ab}^n = (-1)^{\sum a_i b_i}$ . By a divide and conquer approach, if we write a vector in  $\mathbf{R}^{\mathbf{F}_2^n}$  as  $(v, w)$ , where  $v$  and  $w$  split the space in half according to the basis defined above, then

$$H_{n+1}(v, w) = (H_n v + H_n w, H_n v - H_n w)$$

Using this approach, if  $t_n$  is the maximum number of additions and subtractions required to calculate the Walsh transform a  $n$ -dimensional Boolean function, then  $t_{n+1} \leq 2t_n + 2n$ , with  $t_1 = 0$ , hence

$$t_n \leq 2^{n+1} \sum_{k=2}^n k \left(\frac{1}{2}\right)^k \leq n 2^{n+1} \sum_{k=2}^n \left(\frac{1}{2}\right)^k \leq n 2^n$$

so the calculation is possible in  $n2^n$  calculations, which looks bad, but the algorithm is polynomial in the size of the input, because an element of  $\mathbf{R}^{\mathbf{F}_2^n}$  takes  $m = 2^n$  numbers to specify, so the algorithm is  $m \lg m$ . This is essentially the fastest way to compute the Fourier coefficients of an arbitrary boolean function.

## Chapter 2

### Social Choice Functions

We now interpret boolean functions in a particular non-abstract scenario in the theory of social choice. Here a Boolean function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  is employed as a device in a two-candidate election. A group of  $n$  people choose between two candidates, labelled  $-1$  and  $1$ , which are fed into the function  $f$  as a voting rule to determine the overall outcome of the vote. In this situation it is natural to find voting rules with various properties relevant to promoting a fair election, and these properties also have an interest in the general context of the harmonic analysis of Boolean functions.

**Example.** *The standard Boolean function used as a voting rule is the majority function  $\text{Maj}_n : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , which outputs the candidate who got the majority of votes. This is always well defined with an odd number of voters, but there are issues with ties when an even number of voters is given. A function is called a majority function if it always chooses the candidate with the majority of votes where this step is well defined.*

**Example.** *The voting rule  $\text{And}_n : \{-1, 1\}^n \rightarrow \{-1, 1\}$  chooses candidate  $1$  unless every other person votes for candidate  $-1$ . Similarly, the voting rule  $\text{Or}_n : \{-1, 1\}^n \rightarrow \{-1, 1\}$  votes for candidate  $-1$  unless everyone else votes against him.*

**Example.** *The dictator rule  $\chi_i : \{-1, 1\}^n \rightarrow \{-1, 1\}$  defined by  $\chi_i(x) = x_i$  places the power of decision making in a single person.*

**Example.** *All of these functions can be quantified as a version of the weight majority, or linear threshold function, those functions  $f$  which can be defined,*



for some  $a_0, \dots, a_n \in \mathbf{R}$ , as

$$f(x) = \text{sgn}(a_0 + a_1 x_1 + \dots + a_n x_n)$$

Thus each voter has a certain voting power, specified by the  $a_i$ , and there is an additional bias  $a_0$  which pushes the vote in a certain candidate's favour.

**Example.** In the United States, a recursive majority voting rule is used. A simple version of the rule is defined on  $n^d$  bits by

$$\text{Maj}_n^{\otimes d}(x_1, \dots, x_d) = \text{Maj}_n(\text{Maj}_n^{\otimes(d-1)}(x_1), \dots, \text{Maj}_n^{\otimes(d-1)}(x_n))$$

where each  $x_i \in \{-1, 1\}^{n^{d-1}}$ . Thus we subdivide voters into certain regions, determine the majority over this region, and then take the majority over the accumulated choices.

**Example.** The tribes function divides voters into tribes, and the outcome of the vote holds if and only if one tribe is unanimously in favour of the vote. The width  $w$ , size  $s$  tribe voting rule is defined by

$$\text{Tribes}_{ws} : \{-1, 1\}^{sw} \rightarrow \{-1, 1\}$$

$$\text{Tribes}_{ws}(x_1, \dots, x_s) = \text{Or}_s(\text{And}_w(x_1), \dots, \text{And}_w(x_s))$$

where each  $x_i \in \{-1, 1\}^w$ . The number of ways we can screw up each individual election is  $(2^w - 1)$ , so that  $\mathbf{P}(\text{Tribes}_{ws} = 1) = (2^w - 1)^s / 2^{sw} = (1 - 2^{-w})^s$ , and so  $\mathbf{E}[\text{Tribes}_{ws}] = 2(1 - 2^{-w})^s - 1$ . If we control the size of  $w$  and  $s$ , we can therefore get an unbiased education.

Any boolean function is a voting rule on two candidates, so the study of voting functions is really just a language for talking about general properties of boolean functions. However, it certainly motivates the development of certain contexts which will become very useful in the future. For instance, important properties of Boolean functions become desirable properties of voting rules. In particular,

- A voting rule  $f$  is **monotone** if  $x \leq y$  implies  $f(x) \leq f(y)$ .
- A voting rule is **odd** if  $f(-x) = -f(x)$ .
- A rule is **unanimous** if  $f(1) = 1$ , and  $f(-1) = -1$ .

- A voting rule is **symmetric** if  $f(x^\pi) = f(x)$ , where  $\pi \in S_n$  acts on  $\{-1, 1\}^n$  by permuting coordinates.
- A voting rule is **transitive symmetric** if, for any index  $i$ , there is a permutation  $\pi$  taking  $i$  to any other index  $j$ , such that  $f(x^\pi) = f(x)$ .

There is only a single function which satisfies all of these properties for odd  $n$ , the majority function  $\text{Maj}_n$ . This constitutes May's theorem.

**Theorem 2.1.** *The majority function is the only monotone, odd, unanimous, symmetric, transitive symmetric function in odd dimension. In even dimensions, such a function does not exist.*

*Proof.* Suppose  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  is a function which is monotone and symmetric. Then  $f(x) = g(\#\{i : x^i = 1\})$  for some monotone function  $g : \{0, \dots, n\} \rightarrow \{-1, 1\}$ . The monotonicity implies that there is  $N$  for which

$$g(n) = \begin{cases} -1 & n < N \\ 1 & n \geq N \end{cases}$$

Note that if  $f$  is an odd function, then

$$\mathbf{E}[f(X)] = \mathbf{E}[f(-X)] = -\mathbf{E}[f(X)]$$

hence  $\mathbf{E}[f(X)] = 0$ , so  $f$  takes  $+1$  and  $-1$  with equal probability. But also

$$0 = \mathbf{E}[f(X)] = \frac{1}{2^n} \sum_{k=0}^n \binom{n}{k} g(k)$$

Hence

$$\sum_{k=0}^{N-1} \binom{n}{k} = \sum_{k=N}^n \binom{n}{k}$$

Since the left side strictly increases as a function of  $N$ , and the right side strictly decreases, if a  $N$  exists which satisfies this equality it is unique. If  $n$  is odd, then we can pick  $N = (n+1)/2$ , because the identity

$$\binom{n}{k} = \binom{n}{n-k}$$

which implies coefficients can be paired up. If  $n$  is even, the same pairing technique shows that such a choice of  $N$  is impossible.  $\square$

## 2.1 Influence

Given a voting rule  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , we may wish to quantify the power of each voter in changing the outcome of the vote. If we choose a particular voter  $i$  and we fix all inputs but the choice of the  $i$ 'th voter, we say that the voter  $i$  is **pivotal** if his choice affects the outcome of the election. That is, given  $x \in \{-1, 1\}^n$ , we define  $x^{\oplus i} \in \{-1, 1\}^n$  to be the input obtained by flipping the  $i$ 'th bit. Then  $i$  is pivotal on  $x$  exactly when  $f(x^{\oplus i}) \neq f(x^i)$ . It is important to note that even if a voting rule is symmetric, it does not imply that each person has equal power over the results of the election, once other votes have been fixed (For instance, in a majority voting system with 3 votes for candidate  $A$  and 4 votes for candidate  $B$ , the voters for  $B$  have more power because their choice, considered apart from the others seriously impacts the result of the particular election).

It is natural to define the *overall power* of a voter  $i$  on a certain voting rule  $f$  as the likelihood that the voter is pivotal on the election. We call this the **influence** of the voter, denoted  $\text{Inf}_i(f)$ . To be completely accurate, we would have to determine the probability that each particular choice of votes occurs (If there is a high chance that one candidate is going to one, we should expect each voter to individually have little power, whereas if the competition is tight, each voter should have much more power). We assume that all voters choose outcomes independently, and uniformly randomly (this is the **impartial culture** assumption), so that the influence takes the form

$$\text{Inf}_i(f) = \mathbf{P}(f(X) \neq f(X^{\oplus i}))$$

Thus the influence measures the probability that index  $i$  is pivotal uniformly across all elements of the domain.

The impartial culture assumption is not only for simplicity, in the face of no other objective choice of distribution, but also because the formula then has combinatorial interest. If we colour a node on the Hamming cube based on the output of the function  $f$ , then the influence of an index  $i$  is the fraction of coordinates whose color changes when we move along the  $i$ 'th dimension of the cube. Similarly, we can also consider it as the fraction of edges along the  $i$ 'th dimension upon which  $f$  changes.

**Example.** The dictator function  $\chi_i$  satisfies  $\text{Inf}_i(\chi_i) = 1$ , and  $\text{Inf}_j(\chi_i) = 0$  otherwise. If  $f$  is an arbitrary voting rule for which  $\text{Inf}_i(f) = 0$ , then  $f$  completely

ignores index  $i$ , we can actually specify  $f$  as a function of the remaining  $n - 1$  variables.

**Example.** The  $i$ 'th index is pivotal for  $\text{And}_n$  only at the points  $1$  and  $1^{\oplus i}$ , so that  $\text{Inf}_i(\text{And}_n) = 2/2^n = 2^{1-n}$ .

**Example.** To calculate the influence of  $\text{Maj}_n$ , we note that if  $\text{Maj}_n(x) = -1$ , and  $\text{Maj}_n(x^{\oplus i}) = 1$ , then there are  $(n + 1)/2$  indices  $j$  such that  $x^j = -1$ , and  $i$  is one of these indices. Once  $i$  is fixed, the total possible choices of indices in which these circumstances hold is

$$\binom{n-1}{\frac{n-1}{2}}$$

hence if we also consider  $x$  for which  $\text{Maj}_n(x) = 1$ , and  $\text{Maj}_n(x^{\oplus i}) = -1$ , then

$$\text{Inf}_i(\text{Maj}_n) = 2 \binom{n-1}{\frac{n-1}{2}} / 2^n = \binom{n-1}{\frac{n-1}{2}} 2^{1-n}$$

Applying Stirling's approximation, we can compute the approximation

$$\text{Inf}_i(\text{Maj}_n) = \sqrt{\frac{2}{n\pi}} + O(n^{-3/2})$$

We will soon show this is about the best influence we can find for a monotonic, symmetric voting rule.

To connect the influence of a boolean function to its Fourier expansion, we must come up with an analytic expression for the influence. Consider the  $i$ 'th partial derivative operator

$$(D_i f)(x) = \frac{f(x^{i \rightarrow 1}) - f(x^{i \rightarrow -1})}{2}$$

In this equation, we see a slight relation to differentiation of real-valued functions, and the analogy is completed when we see how  $D_i$  acts on the polynomial representation of  $f$ . Indeed,

$$D_i x^S = \begin{cases} x^{S - \{i\}} & : i \in S \\ 0 & : i \notin S \end{cases}$$

Hence by linearity,

$$(D_i f)(x) = \sum_{i \in S} \hat{f}(S) x^{S - \{i\}}$$

For Boolean-valued functions  $f$ ,  $D_i f$  is intimately connected to the influence of  $f$ , because

$$(D_i f)(x) = \begin{cases} \pm 1 & f(x^i) \neq f(x^{\oplus i}) \\ 0 & f(x^i) = f(x^{\oplus i}) \end{cases}$$

Hence  $(D_i f)^2(x)$  is the 0-1 indicator of whether  $i$  is pivotal at  $x$ , and so

$$\text{Inf}_i(f) = \mathbf{E}[(D_i f)^2] = \|D_i f\|_2^2 = \sum_{i \in S} \hat{f}(S)^2$$

Defining the  $i$ 'th **Laplacian** operator  $L_i = x_i D_i$ , we find

$$(L_i f)(x) = \sum_{i \in S} \hat{f}(S) x^S$$

Hence  $L_i$  is a projection operator, satisfying  $\text{Inf}_i(f) = \langle L_i f, L_i f \rangle = \langle f, L_i f \rangle$ . Thus we have found a quadratic representation of  $\text{Inf}_i(f)$ . This will have widespread influences throughout the theory. In particular, we can now already see that an index  $i$  is nonessential if and only if  $\hat{f}(S) = 0$  for all sets  $S$  such that  $i \in S$ .

We shall dwell on the Laplacian for slightly longer than the other operators, since we will find its generalization occurring in future discussion. As a projection operator, it has a complementary projection

$$(E_i f)(x) = \frac{f(x^{i \rightarrow 1}) + f(x^{i \rightarrow -1})}{2}$$

which can be considered the expected value of  $f$ , if we fix all coordinates except for the  $i$ 'th index, which takes a uniformly random value. Thus for any boolean function  $f$  we have

$$f = E_i f + x_i D_i f$$

Both  $E_i f$  and  $D_i f$  do not depend on the  $i$ 'th coordinate of  $f$ , which is useful for carrying out inductions on the dimension of a boolean function's

domain. We also note that we have a representation of the Laplacian as a difference

$$(L_i f)(x) = \frac{f(x) - f(x^{\oplus i})}{2}$$

so that the Laplacian acts very similarly to  $D_i$ .

Though we use the fact that  $(D_i f)^2 = |D_i f|$  to obtain a quadratic equation for  $D_i f$ , the definition of the influence as the absolute value  $|D_i f|$  as the definition of influence still has its uses, especially when obtaining  $L^1$  inequalities for the influence. Indeed, we can establish the bound

$$\text{Inf}_i(f) = \mathbf{E}|D_i f| \geq |\mathbf{E}(D_i f)| = |\hat{f}(i)|$$

This inequality only becomes an equality when  $f$  is always increasing in the  $i$ 'th direction, or always decreasing – that is, if  $D_i f \geq 0$  or  $D_i f \leq 0$ , and if  $D_i f \geq 0$  we in fact have  $\text{Inf}_i(f) = \hat{f}(i)$ , since  $\hat{f}(i)$  is non-negative. In particular, if  $f$  is monotone, then we have  $\text{Inf}_i(f) = \hat{f}(i)$  for all  $i$ , which is very useful. As a first application of this fact, we show that in a monotone, transitive symmetric voting system, all voters have small influence.

**Theorem 2.2.** *Given a monotone, transitive symmetric  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ ,*

$$\text{Inf}_i(f) \leq \frac{1}{\sqrt{n}}$$

*Proof.* Applying Parseval's inequality, we find

$$1 = \|f\|_2^2 = \sum \hat{f}(S)^2 \geq \sum \hat{f}(i)^2 = n \hat{f}(i)^2$$

Hence  $\hat{f}(i) \leq 1/\sqrt{n}$ . But by monotonicity,

$$\text{Inf}_i(f) = \hat{f}(i)$$

Putting these equations together gives us the inequality.  $\square$

The second is that there are very few Boolean valued functions with Fourier coefficients bounded by degree one.

**Lemma 2.3.** *If  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  has  $W^{\leq 1}(f) = 1$ , then  $f$  is either constant, or  $\pm x^i$  for some  $i \in [n]$ .*

*Proof.* Write  $f(x) = a + \sum b_j x_j$ . Then  $(D_i f)(x) = b_i$ . Because  $f$  is Boolean valued,

$$(D_i f)(x) = \frac{f(x^{i \rightarrow 1}) - f(x^{i \rightarrow -1})}{2} \in \{0, \pm 1\}$$

Hence  $b_i = 0$  or  $b_i = \pm 1$ . If all  $b_i$  are equal to zero, then  $f$  is constant. Otherwise, there is  $b_i$  with  $b_i^2 = 1$ , and then all other coefficients must vanish, and so  $f(x) = x_i$  or  $f(x) = -x_i$ .  $\square$

The **total influence** of a boolean function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , denoted  $\mathbf{I}(f)$ , is the sum of the influences  $\text{Inf}_i(f)$  over each coordinate. It is not a measure of how powerful each voter is, but instead how chaotic the system is with respect to how any of the voters change their coordinates. For instance, constant functions have total influence zero, where no vote change effects the outcome in any way, whereas the function  $x^{[n]}$  has total influence  $n$ , since every change in a voters choices flips the outcome of the vote. For monotone transitive symmetric voting rules, we already know the total influence is bounded by  $\sqrt{n}$ , and this is asymptotically obtained by the majority function.

**Example.** The total influence of  $\chi_S$  is  $|S|$ , because all the indices of  $S$  are pivotal for all inputs. The total influence is maximized over all functions by the parity function  $x^{[n]}$ , which is always pivotal for all inputs. The total influence of  $\text{And}_n$  and  $\text{Or}_n$  is  $n2^{1-n}$ , rather small, whereas  $\text{Maj}_n$  has total influence

$$\sqrt{2n/\pi} + O\left(\sqrt{1/n}\right)$$

which is the maximum total influence of any monotone transitive symmetric function.

**Example.** The tribes function

As a sum of quadratic forms, the total influence is also a quadratic form, but cannot be represented by a projection. Considering

$$\mathbf{I}(f) = \sum_i \text{Inf}_i(f) = \sum_i \langle f, L_i f \rangle = \left\langle f, \left( \sum_i L_i \right) f \right\rangle$$

we see that the total influence is represented as a quadratic form over the **Laplacian** operator  $L = \sum L_i$ . Note that

$$(Lf)(x) = \sum_{i=1}^n \sum_{j \in S} \hat{f}(S) x^S = \sum |S| \hat{f}(S) x^S$$

so the Laplacian amplifies the coefficients of  $f$  corresponding to sets of large weight. This makes sense, because  $x^S$  changes on more inputs when  $S$  contains more indices, hence the function should have higher influence. This implies that

$$\mathbf{I}(f) = \sum |S| \hat{f}(S)^2 = \sum k \mathbf{W}^k(f)$$

for Boolean-valued functions, the total influence is the expected cardinality of  $\mathcal{S}$ , where  $\mathcal{S}$  is a random set distributed according to the spectral distribution induced from  $f$ .

There is a probabilistic interpretation of the total influence, which makes it interesting to the theory of voting systems. Define  $\text{sens}_f(x)$  to be the number of pivotal indices for  $f$  at  $x$ . Then  $\mathbf{I}(f) = \mathbf{E}[\text{sens}_f(X)]$ . Indeed, we find

$$(Lf)(x) = \sum_i \frac{f(x) - f(x^{\oplus i})}{2} = \frac{n}{2} [f(x) - \mathbf{E}[f(x^{\oplus i})]]$$

where the expectation is taken where  $i$  is random, uniformly distributed over all indices  $[n]$ . We have

$$\mathbf{E}[f(x^{\oplus i})] = \frac{n - \text{sens}_f(x)}{n} f(x) + \frac{\text{sens}_f(x)}{n} (-f(x)) = f(x) \frac{n - 2\text{sens}_f(x)}{n}$$

Hence  $(Lf)(x) = f(x) \text{sens}_f(x)$ .

There is also a combinatorial interpretation of the total influence. Each vertex has  $n$  edges, and if we consider the probability distribution which first picks a point on the cube uniformly, and then an edge uniformly, then the resulting distribution will be the uniform distribution on edges. Since the expected number of **boundary edges** (those  $(x, y)$  for which  $f(x) \neq f(y)$ ) emanating from a vertex is  $\mathbf{I}(f)$ , the chance that the edge we pick will be a boundary edge is  $\mathbf{I}(f)/n$ .

Since  $\mathbf{V}(f) = \sum_{k \neq 0} \mathbf{W}^k(f)$ , and  $\mathbf{I}(f) = \sum_{k \neq 0} k \mathbf{W}^k(f)$ , it is fairly trivial to obtain the Poincare inequality  $\mathbf{V}(f) \leq \mathbf{I}(f)$ , which gives a strict inequality unless  $\mathbf{W}^{>2} f = 0$ . If the low degree coefficients of the function vanish, then we can give an even sharper inequality – if  $\mathbf{W}^{<k}(f) = 0$ , then  $\mathbf{V}(f) \leq \mathbf{I}(f)/k$ . Conversely,  $\text{Inf}_i(f) \leq \mathbf{V}(f) \leq \mathbf{I}(f)$ , so the variance is likely close to the total influence when the influence is concentrated on a single voter. For Boolean valued functions, the only such functions are the singular characters  $\pm x^i$  and the constant functions. Geometrically, the Poincare



inequality gives an edge expansion bound for functions on the Hamming cube. Given a subset  $A$  of the  $n$  dimensional Hamming cube with  $m$  points, and if  $\alpha = m/2^n$ , then the ‘characteristic function’  $I_f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  has variation  $4\alpha(1 - \alpha)$ , and therefore the fraction of edges in the cube which form a boundary edge is lower bounded by  $4\alpha(1 - \alpha)/n$ . In particular, for the set  $A$  corresponding to  $x^{[n]}$ , where  $\alpha = 1/2$ , we see this bound is tight, as for  $\alpha = 0$  and  $\alpha = 1$ . However, for other values of  $\alpha$  must better bounds can be obtained.

**Lemma 2.4.** *If  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  is an unbiased Boolean function, then there is  $i$  with  $\text{Inf}_i(f) \geq 2/n - 4/n^2$ .*

*Proof.* Let  $j$  be the index which maximizes  $\text{Inf}_j(f)$ . First, note that  $\text{Inf}_j(f) \geq 1/n$ , for if  $\text{Inf}_i(f) < 1/n$ , then we find  $\mathbf{I}(f) < 1 = \mathbf{V}(f)$ , which is impossible. Now, because  $f$  is unbiased,

$$\mathbf{I}(f) \geq \mathbf{W}^1(f) + 2(1 - \mathbf{W}^1) = 2 - \mathbf{W}^1(f)$$

Now we use the fact that  $|\hat{f}(i)| \leq \text{Inf}_i(f)$  to conclude that

$$\mathbf{W}^1(f) = \sum \hat{f}(i)^2 \leq n \text{Inf}_j(f)^2$$

hence  $n \text{Inf}_j(f) \geq \mathbf{I}(f) \geq 2 - n \text{Inf}_j(f)^2$ . Rearranging, we can calculate that  $\text{Inf}_j(f) \geq 2/n - \text{Inf}_j(f)^2$ , hence the inequality is satisfied assuming that  $\text{Inf}_j(f) \leq 2/n$ . Thus we can assume that  $\text{Inf}_j(f) \geq 2/n$ , but then the inequality is trivial.  $\square$

Returning to our discussion of voting systems, an additional property to consider is the expected number of voters who agree with the outcome of the vote.

**Theorem 2.5.** *Let  $f$  be a voting rule for a 2-candidate election. If  $w(x)$  is the number of  $x^i$  which agree with the voting choice, then*

$$\mathbf{E}(w) = \frac{n}{2} + \frac{1}{2} \sum \hat{f}(i)$$

*which is maximized by the majority functions.*

*Proof.* We have

$$w(x) = \sum_{i=1}^n \frac{1 + f(x)x^i}{2}$$

Hence

$$\mathbf{E}(w(X)) = \frac{n}{2} + \frac{1}{2} \sum_{i=1}^n \mathbf{E}[f(x)x^i] = \frac{n}{2} + \frac{1}{2} \sum_{i=1}^n \hat{f}(i)$$

We write

$$\sum_{i=1}^n \hat{f}(i) = \mathbf{E}(f(X)(X^1 + X^2 + \dots + X^n)) \leq \mathbf{E}(|X^1 + X^2 + \dots + X^n|)$$

and this inequality turns into an equality precisely when  $f$  has the property that  $f(X) = \text{sgn}(\sum X^i)$ , whenever  $\sum X^i \neq 0$ .  $\square$

**Example.** Consider the function  $f$  chosen randomly from all Boolean valued functions on  $\{-1, 1\}^n$ . We calculate

$$\mathbf{E}[\text{Inf}_i(f)] = \mathbf{E}_X[\mathbf{P}(f(X) \neq f(X^{\oplus i}))] = \mathbf{E}_X[1/2] = 1/2$$

Hence  $\mathbf{E}[\mathbf{I}(f)] = n/2$ , so for an average function half of the voters agree with the outcome.

## 2.2 Noise and Stability

In realistic voting systems, the inputs to voting systems are rarely the most reliable. Some voters don't even show up to voting booths to announce their choice of candidate, and errors in the recording of data mean that the input might not even be accurate to the choices of voters. It is important for a voting rule to remain stable under these perturbations, so that the outcome of the modified vote is likely to be the same as the outcome of the original vote, if we had perfect knowledge of all voting choices.

So our general rule with which we will define the **stability** of a voting rule  $f$ , is to find the probability that  $f(x) \neq f(y)$ , where  $y$  is obtained by a slight perturbation of  $x$ . We'll say two Boolean-valued random variables  $X$  and  $Y$  are  $\rho$ -correlated, for some  $\rho \in [-1, 1]$  if  $\mathbf{E}[XY] = \rho$ . This is equivalent to

$$\mathbf{P}(Y = X) = \frac{1 + \rho}{2}$$

Given a random variable  $X$ , we can generate a random variable  $Y$  which is  $\rho$ -correlated to  $X$  by letting  $Y = X$  with probability  $\rho$ , and otherwise choosing  $Y$  uniformly randomly. We write  $Y \sim N_\rho(X)$  if  $Y$  is  $\rho$ -correlated to  $X$ . This is the sense with which we will need  $\rho$ -correlation, for  $\rho$ -correlation represents a certain value being randomly perturbed by a small quantity. Similarly, if  $X$  and  $Y$  are multidimensional Boolean-valued functions, we say  $X$  is  $\rho$ -correlated to  $Y$  if the coordinates of  $X$  and  $Y$  are independent, and each coordinate in  $X$  is  $\rho$ -correlated to the corresponding coordinate in  $Y$ . We can now define the stability with respect to  $\rho$  as

$$\text{Stab}_\rho(f) = \mathbf{E}_{Y \sim N_\rho(X)}[f(X)f(Y)]$$

where  $X$  is chosen uniformly. We of course find

$$\text{Stab}_\rho(f) = 2 \left( \mathbf{P}_{Y \sim N_\rho(X)}[f(X) = f(Y)] \right) - 1$$

Thus if the voting system is modified by some small random quantity  $\rho$ , the chance that this will affect the outcome of the vote is  $[1 + \text{Stab}_\rho(f)]/2$ .

An alternate measure of the sensitivity of a function, which may be easier to reason with, is the probability that the vote is disrupted if we purposefully reverse each coordinate of  $X$  independently with a small probability  $\delta$ , forming the random variable  $Y$ . Thus  $\mathbf{P}(Y = X) = 1 - \delta$ , hence  $Y$  is  $1 - 2\delta$ -correlated to  $X$ . We define the **noise sensitivity** based on this correlation to be

$$\mathbf{NS}_\delta(f) = \mathbf{P}[f(X) \neq f(Y)]$$

Since  $Y$  is  $1 - 2\delta$  correlated with  $X$ , and by our previous formula, we find

$$\mathbf{NS}_\delta(f) = \frac{1 - \text{Stab}_{1-2\delta}(f)}{2}$$

So that there is a linear correlation between the two measures of stability.

**Example.** The constant functions  $\pm 1$  have stability 1, since there is no chance of changing the value of the functions. The dictator functions  $x^i$  satisfy  $\mathbf{NS}_\delta(x^i) = \delta$ , hence

$$\text{Stab}_\rho(x^i) = 1 - 2\mathbf{NS}_{\frac{1-\rho}{2}}(x^i) = \rho$$

More generally, the stability of  $x^S$  is

$$\text{Stab}_\rho(x^S) = \mathbf{E}(X^S Y^S) = \prod_{i \in S} \mathbf{E}(X^i Y^i) = \prod_{i \in S} \left( \left( \frac{1+\rho}{2} \right) - \left( \frac{1-\rho}{2} \right) \right) = \rho^{|S|}$$

The noise stability of the majority functions has no convenient formula, but it does tend to a nice limit at the number of voters increases ad infimum. Later on, we will prove that for  $\rho \in [-1, 1]$ ,

$$\lim_{\substack{n \rightarrow \infty \\ n \text{ odd}}} \text{Stab}_\rho(\text{Maj}_n) = (2/\pi) \sin^{-1}(\rho)$$

Hence for  $\delta \in [0, 1]$ ,

$$\lim_{\substack{n \rightarrow \infty \\ n \text{ odd}}} \text{NS}_\delta[\text{Maj}_n] = (1/\pi) \arccos(1 - 2\delta) = \frac{2\sqrt{\delta}}{\pi} + O(\delta^{3/2})$$

Hence the noise stability curves sharply near small and large values of  $\rho$ .

That noise stability is tightly connected to the Fourier coefficients of the function is one of the most powerful tools in Boolean harmonic analysis. As with the influence, we introduce an operator to provide us with a connection to the coefficients. We introduce the **noise operator** with parameter  $\rho$  as

$$(T_\rho f)(x) = \mathbf{E}_{X \sim N_\rho(x)} f(X)$$

which ‘smoothes out’ the function  $f$  by a certain parameter  $\rho$ .  $T_\rho$  is obviously linear, and

$$(T_\rho x^S) = \mathbf{E}[X^S] = \prod_{i \in S} \mathbf{E}[X^i] = \prod_{i \in S} \left( \left( \frac{1+\rho}{2} \right) x^i - \left( \frac{1-\rho}{2} \right) x^i \right) = \rho^{|S|} x^S$$

Hence the noise operator transforms the Fourier coefficients as

$$(T_\rho f)(x) = \sum \rho^{|S|} \hat{f}(S) x^S$$

The connection between the noise operator and the stability is that

$$\begin{aligned} \text{Stab}_\rho(f) &= \mathbf{E}[f(X)f(Y)] = \mathbf{E}[f(X)\mathbf{E}[f(Y)|X]] \\ &= \mathbf{E}[f(X)(T_\rho f)(X)] = \langle f, T_\rho f \rangle \end{aligned}$$

Thus stability is a quadratic form, and so we find

$$\text{Stab}_\rho(f) = \sum_S \rho^{|S|} \widehat{f}(S)^2 = \sum_k \rho^k \mathbf{W}^k(f)$$

Correspondingly, we have

$$\begin{aligned} \mathbf{NS}_\delta(f) &= \frac{1 - \text{stab}_{1-2\delta}(f)}{2} \\ &= \frac{1 - \mathbf{W}^0(f)}{2} - \sum_{k \neq 0} \frac{(1 - 2\delta)^k}{2} \mathbf{W}^k(f) \\ &= \frac{1}{2} \sum (1 - (1 - 2\delta)^k) \mathbf{W}^k(f) \end{aligned}$$

Hence  $\mathbf{NS}_\delta$  can also be represented as a quadratic form.

That all the operators that we have found can be considered as a quadratic form should not be taken as a hint that all interesting operators in Boolean analysis are quadratic, just that the ones easiest to understand happen to be. And these operators are even more easy to understand because they are diagonalized by the characters  $x^S$ , which all happen to be Eigenvalues. And if that wasn't enough, all the eigenvalues are positive, which implies that the functionals are convex – given a functional  $F$  with  $F(\sum a_S x^S) = \sum b_S a_S^2$ , where  $b_S \geq 0$ , and given  $f$  and  $g$ , using the convexity of  $x^2$  we find

$$\begin{aligned} F(\lambda f + (1 - \lambda)g) &= \sum_S b_S \left[ \lambda \widehat{f}(S) + (1 - \lambda) \widehat{g}(S) \right]^2 \\ &\leq \sum_S b_S \left[ \lambda \widehat{f}(S)^2 + (1 - \lambda) \widehat{g}(S)^2 \right] \\ &\leq \lambda F(f) + (1 - \lambda) F(g) \end{aligned}$$

Thus these operators are incredibly applicable to understanding.

The operators  $T_\rho$  form a local one-parameter semigroup of the set of all operators on Boolean functions, because  $T_\rho T_\mu = T_{\rho\mu}$ . This essentially follows because if  $X$  is  $\mu$  correlated to the constant  $x \in \{-1, 1\}$ , and  $Y$  is  $\rho$  correlated to  $X$ , then  $Y$  is  $\rho\mu$  correlated to  $x$ , since

$$\begin{aligned} \mathbf{P}(Y = x) &= \mathbf{P}(X = x) \mathbf{P}(Y = X) + \mathbf{P}(X \neq x) \mathbf{P}(Y \neq X) \\ &= \left( \frac{1 + \mu}{2} \right) \left( \frac{1 + \rho}{2} \right) + \left( \frac{1 - \mu}{2} \right) \left( \frac{1 - \rho}{2} \right) = \frac{1 + \rho\mu}{2} \end{aligned}$$

and this completes the argument. What's more,  $T_\rho$  is a contraction on  $L^q\{-1,1\}^n$  for all  $q \geq 1$ , because by Jensen's inequality, since  $x \mapsto x^q$  is convex,

$$\begin{aligned}\|T_\rho f\|_q^q &= \mathbf{E}[(T_\rho f)^q(X)] \\ &= \mathbf{E}[\mathbf{E}_{Y \sim N_\rho(X)}[f(Y)]^q] \\ &\leq \mathbf{E}[\mathbf{E}_{Y \sim N_\rho(X)}[f^q(Y)]]\end{aligned}$$

If  $X$  is chosen uniformly randomly, and  $Y \sim N_\rho(X)$ , then  $Y$  is also chosen uniformly randomly, since

$$\begin{aligned}\mathbf{P}(Y = 1) &= \mathbf{P}(Y = X)\mathbf{P}(X = 1) + \mathbf{P}(Y \neq X)\mathbf{P}(X = -1) \\ &= \frac{\mathbf{P}(Y = X) + \mathbf{P}(Y \neq X)}{2} = \frac{1}{2}\end{aligned}$$

Hence

$$\mathbf{E}[\mathbf{E}_{Y \sim N_\rho(X)}[f^q(Y)]] = \mathbf{E}[f^q(X)] = \|f\|_q^q$$

and we may obtain the inequality by taking  $q$ 'th roots.

Using the Fourier representation, we can show that the dictator functions maximize stability among the unbiased Boolean-valued functions.

**Theorem 2.6.** *For  $\rho \in (0, 1)$ , if  $f$  is an unbiased Boolean-valued function, then  $\text{Stab}_\rho(f) \leq \rho$ , with equality if and only if  $f$  is a dictator.*

*Proof.* We write

$$\text{Stab}_\rho(f) = \sum \rho^{|S|} \hat{f}(S)^2 \leq \rho \sum \hat{f}(S)^2 = \rho$$

If this inequality is an equality, then  $\mathbf{W}^{>1}(f) = 0$ , and it then follows that  $f$  is either constant or a dictator function, and the constant functions are biased.  $\square$

Notice that  $\text{Stab}_\rho(f)$  is a polynomial in  $\rho$ , with non-negative coefficients. It follows that the stability of a function increases as  $\rho$  increases for positive  $\rho$ , and  $\text{Stab}_0(f) = 1$ . What's more, we find that, looking at the coefficients, that

$$\frac{d\text{Stab}_\rho(f)}{d\rho} = \sum_{S \neq \emptyset} |S| \rho^{|S|-1} \hat{f}(S)^2$$

Hence the derivative at  $\rho = 0$  is  $\mathbf{W}^1(f)$ , and the derivative at  $\rho = 1$  is  $\mathbf{I}(f)$ . We can also use this equation with the mean value theorem to determine a useful bound.

**Theorem 2.7.** *For any Boolean function  $f$ ,*

$$|\text{Stab}_\rho(f) - \text{Stab}_{\rho-\varepsilon}(f)| \leq \frac{\varepsilon}{1-\rho} \mathbf{V}(f)$$

*Proof.* Using the mean value theorem, we can find  $\eta \in (\rho - \varepsilon, \rho)$  such that

$$\text{Stab}_\rho(f) - \text{Stab}_{\rho-\varepsilon}(f) = \varepsilon \frac{d\text{Stab}_\eta(f)}{d\eta} = \varepsilon \sum k\eta^{k-1} \mathbf{W}^k(f)$$

Now we use the fact that  $k\eta^{k-1} \leq 1/(1-\eta)$  for all  $0 \leq \eta \leq 1$ , hence

$$\varepsilon \sum k\eta^{k-1} \mathbf{W}^k(f) \leq \frac{\varepsilon}{1-\eta} \mathbf{V}(f) \leq \frac{\varepsilon}{1-\rho} \mathbf{V}(f)$$

Hence we have a Lipschitz inequality for  $\text{Stab}_\rho(f)$  which explodes at the boundary, because the estimate  $\sum x^k = 1/(1-x)$  we used to establish this estimate breaks down at  $\rho = 1$ .  $\square$

We can incorporate stability into the concept of influence. For  $\rho \in [0, 1]$ , define the  **$\rho$ -stable influence**

$$\text{Inf}_i^\rho(f) = \text{Stab}_\rho(D_i f) = \sum_{S \in \mathcal{S}} \rho^{|S|-1} \hat{f}(S)^2$$

The **total  $\rho$ -stable influence** is

$$\mathbf{I}^\rho(f) = \sum_{i \in S} \text{Inf}_i^\rho(f) = \sum_{i \in S} |S| \rho^{|S|-1} \hat{f}(S)^2$$

In this form, we see that  $\mathbf{I}^\rho(f) = d\text{Stab}_\rho(f)/d\rho$ , so that this  $\rho$ -stable influence measures the change in stability of a function with respect to  $\rho$ . There isn't too much of a natural interpretation of the  $\rho$ -stable influences. Since  $D_i f$  measures if  $f$  is pivotal at  $f$ , one intuitive insight about the  $\rho$ -stable influence is that it is a way of smoothening out how pivotal  $D_i f$  is; Though  $D_i f$  only measures the change in  $f$  over the  $i$ 'th coordinate, the  $\rho$ -stable influence now incorporates a slight change in the other coordinates as well. Regardless of their interpretation, they will be technically very useful later on.

**Theorem 2.8.** Suppose that a function  $f$  satisfies  $\mathbf{V}(f) \leq 1$ . If  $0 < \delta, \epsilon < 1$ , then the number of indices  $i$  for which  $\text{Inf}_i^{(1-\delta)}(f) \geq \epsilon$  is less than or equal to  $1/\delta\epsilon$ .

*Proof.* Let  $J$  be the set of indices for which  $\text{Inf}_i^{(1-\delta)}(f) \geq \epsilon$ , so that we must prove  $|J| \leq 1/\delta\epsilon$ . We obviously have  $|J| \leq \mathbf{I}^{(1-\delta)}(f)/\epsilon$ , hence we need only prove  $\mathbf{I}^{(1-\delta)}(f) \leq \frac{1}{\delta}$ . Since

$$\begin{aligned} \mathbf{I}^{(1-\delta)}(f) &= \sum |S|(1-\delta)^{|S|-1} \hat{f}(S)^2 \\ &\leq \max(|S|(1-\delta)^{|S|-1}) \sum \hat{f}(S)^2 = \max(|S|p^{|S|-1}) \end{aligned}$$

Hence it suffices to prove that  $k\delta(1-\delta)^{k-1} \leq 1$  for any  $0 < \delta < 1$  and  $k \geq 1$ . Note that for  $0 < x < 1$ ,

$$nx^n \leq \sum_{k=0}^n x^k \leq \sum_{k=0}^{\infty} x^k = 1/(1-x)$$

Our inequality follows by setting  $x = 1 - \delta$ . □

## 2.3 Arrow's Theorem

The majority system is perfect for a two-candidate voting system. It is monotone, symmetric, and maximizes the number of voters which agree with each choice. But for three-candidate preferential voting system, we have no information. Given a series of rankings of  $n$  candidates, we desire a way to decide upon a winner. In 1785, the French philosopher and mathematician Nicolas de Condorcet suggested breaking down individual voting preferences of voters into pairwise choices over candidates. Once these candidates are considered pairwise, we can then evaluate which candidate is the best over all comparisons. Thus we require a certain voting rule  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  to consider candidates pairwise, to consider such **Condorcet elections**. In this section, we consider only 3-candidate scenarios.

For instance, consider the Condorcet election over 3 candidates  $a$ ,  $b$ , and  $c$  using the majority function as a voting rule. Consider three voters, with preferences  $a > b > c$ ,  $a > c > b$ , and  $b > c > a$ . We see that  $b$  is favoured more often than  $c$ , whereas  $a$  is favoured more often than both



$b$  and  $c$ , hence  $a$  is the overall winner of the election. We say  $a$  is the **Condorcet winner**. Given a particular voter, we define the  $xy$  choice of the voter to be the element of  $\{-1, 1\}$ , where 1 indicates the voter favours  $x$ , and  $-1$  favours  $y$ . We can identify a particular choice of preference of a voter by consider their  $ab$  preference, their  $bc$  preference, and their  $ca$  preference, which is an element of  $\{-1, 1\}^3$ , and conversely, an element of  $\{-1, 1\}^3$  gives rise to a unique linear preference, provided that not all of the coordinates of the vector are equal, that is, the vector isn't  $\{-1, -1, -1\}$  or  $\{1, 1, 1\}$ .

Unfortunately, Condorcet discovered that his election system may not lead to a definite winner. Consider the preferences  $c > a > b$ ,  $a > b > c$ , and  $b > c > a$ , using the Condorcet method. Then  $a$  is preferred to  $b$ ,  $b$  is preferred to  $c$ , and  $c$  is preferred to  $a$ , so there is no definite winner. In particular, if we consider the winner of each election as a tuple of  $ab$ ,  $bc$ , and  $ca$  preferences in  $\{-1, 1\}^3$ , then we can reconstruct a Condorcet winner if and only if the coordinates of the preference vector aren't  $\{-1, -1, -1\}$  or  $\{1, 1, 1\}$ . Thus the majority rule appears to be deficient in a preferential election.

It was Kenneth Arrow in the 1950s who found that there is always a chance that a Condorcet winner does not occur with *any* voting rule to decide upon a pairwise Condorcet winner, except for a particular unattractive option: the dictator function. In 2002, Kalai gave a new proof of this theorem using the tools of Fourier analysis. As should be expected from our analysis, it looks at the properties of the 'not all equal' function  $\text{NAE} : \{-1, 1\}^3 \rightarrow \{0, 1\}$ . Instead of looking at a particular element of the domain of the function to determine whether a Condorcet winner is not always possible, Kalai instead computes the *probability* of a Condorcet winner of a voting rule, where the preferences are chosen over all possibilities.

**Lemma 2.9.** *The probability of a Condorcet winner in a Condorcet election using the pairwise voting rule  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  is precisely*

$$\frac{3}{4}[1 - \text{Stab}_{-1/3}(f)]$$

*Proof.* Let  $x, y, z \in \{-1, 1\}^n$  be chosen such that  $(x_i, y_i, z_i)$  gives the  $(ab, bc, ca)$  preferences of a voter  $i$ . The  $x, y, z$  are chosen uniformly, in such a way that each  $(x_i, y_i, z_i)$  is chosen independently, and uniformly over the 6 tuples

which satisfy the not all equal predicate. There is a Condorcet winner if and only if  $\text{NAE}(f(x), f(y), f(z)) = 1$ , hence

$$\mathbf{P}(f \text{ gives a Condorcet winner}) = \mathbf{E}[\text{NAE}(f(X), f(Y), f(Z))]$$

and since NAE has the Fourier expansion

$$\text{NAE}(x, y, z) = \frac{3 - xy - xz - yz}{4}$$

Hence

$$\mathbf{E}[\text{NAE}(f(X), f(Y), f(Z))] = \frac{3}{4} - \frac{\mathbf{E}(f(X)f(Y)) + \mathbf{E}(f(Y)f(Z)) + \mathbf{E}(f(Z)f(X))}{4}$$

Finally, note that each coordinate of  $X, Y, Z$  is independant, and

$$\mathbf{E}[XY] = \mathbf{E}[YZ] = \mathbf{E}[ZX] = 2/6 - 4/6 = -1/3$$

Thus  $(X, Y)$ ,  $(Y, Z)$ , and  $(Z, X)$  are  $-1/3$  correlated, hence

$$\mathbf{E}[\text{NAE}(f(X), f(Y), f(Z))] = (3/4)(1 - \text{stab}_{-1/3}(f))$$

and this gives the required formula.  $\square$

**Example.** Using the majority function  $\text{Maj}_n$ , we find that the probability of a Condorcet winner tends to

$$\frac{3}{4}(1 - (2/\pi)\sin^{-1}(-1/3)) = \frac{3}{2\pi}\cos^{-1}(-1/3)$$

thus there is approximately a 91.2% chance of a Condorcet winner occuring. This is Guilbaud's formula.

**Theorem 2.10** (Kalai). *The only voting rule  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  which always gives a Condorcet winner is the dictator functions  $x^i$ , or its negation.*

*Proof.* If  $f$  always gives a Condorcet winner, the chance of a winner is 1, hence rearranging the formula just derived, we find  $\text{Stab}_{-1/3}(f) = -1/3$ . But then

$$\sum (-1/3)^k \mathbf{W}^k(f) = -1/3$$

Since  $(-1/3)^k > -1/3$  for  $k > 1$ , we find

$$\sum (-1/3)^k \mathbf{W}^k(f) \geq -1/3 \sum \mathbf{W}^k(f) = -1/3$$

and this inequality becomes an equality if and only if  $\mathbf{W}^{>1}(f) = 0$ , which we have verified implies that  $f$  is constant, a dictator function, or its negation. Assuming  $f$  is unanimous, we find that  $f$  must be the dictator function.  $\square$

We might wonder which voting rule gives us the largest chance of a Condorcet winner. It turns out that we can only obtain a slightly more general result than for the majority function.

**Lemma 2.11.** *If  $f : \{-1, 1\} \rightarrow \{-1, 1\}$  has all  $\hat{f}(i)$  equal, then*

$$\mathbf{W}^1(f) \leq 2/\pi + O(1/n)$$

*Proof.* Since  $n\hat{f}(i) = \sum \hat{f}(i) \leq \sqrt{2n/\pi} + O(\sqrt{1/n})$  (the majority function maximizes  $\sum \hat{f}(i)$ ), we find

$$\mathbf{W}^1(f) = n\hat{f}(i)^2 = \frac{1}{n}(n\hat{f}(i))^2 \leq 2/\pi + O(1/n)$$

Hence the inequality holds.  $\square$

**Theorem 2.12.** *In a 3-candidate Condorcet election with  $n$  voters, the probability of a Condorcet winner is at most  $(7/9) + (2/9)\mathbf{W}^1(f)$*

*Proof.* Given any  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , the probability of a Condorcet has a Fourier expansion

$$\begin{aligned} \frac{3}{4} - \frac{3}{4}\text{Stab}_{-1/3}(f) &= \frac{3}{4} - \frac{3}{4} \sum (-1/3)^k \mathbf{W}^k(f) \\ &= \left( \frac{3}{4} - \mathbf{E}[f] \right) + \frac{\mathbf{W}^1(f)}{4} - \frac{\mathbf{W}^2(f)}{12} + \frac{\mathbf{W}^3(f)}{36} - \dots \\ &\leq (3/4) + \frac{\mathbf{W}^1(f)}{4} + \frac{1}{36} \left( \sum_{k=2}^n \mathbf{W}^k(f) \right) \\ &\leq (3/4) + \frac{\mathbf{W}^1(f)}{4} + \frac{1 - \mathbf{W}^1(f)}{36} = 7/9 + 2/9\mathbf{W}^1(f) \end{aligned}$$

$\square$

Combining these two results, we can conclude that in any voting system with all  $\hat{f}(i)$  equal (which is certainly true even in a transitive symmetric system), the chance of a Condorcet winner is

$$(7/9) + (2/9)\mathbf{W}^1(f) \leq (7/9) + (4/9\pi) + O(1/n)$$

And this is approximately 91.9%. Later on, we will be able to show that this bound holds given a much weaker hypothesis, and what's more, we will show that the majority function actually maximizes the probability.

The advantage of Kalai's approach is that we can use it to obtain a much more robust result, ala linearity testing.

**Theorem 2.13.** *If the probability of a Condorcet election of  $1 - \varepsilon$ , then  $f$  is  $O(\varepsilon)$  close to being a dictator, or the negation of a dictator.*

*Proof.* Using the bound we just used, we find  $1 - \varepsilon \leq 7/9 + 2/9\mathbf{W}^1(f)$ , hence

$$1 - (9/2)\varepsilon \leq \mathbf{W}^1(f)$$

and then the result follows from the Friedgut Kalai Naor theorem.  $\square$

**Theorem 2.14** (Friedgut-Kalai-Naor theorem). *If  $f$  is a Boolean valued function, and  $\mathbf{W}^1(f) \geq 1 - \delta$ , then  $f$  is  $O(\delta)$  close to  $\pm x^i$  for some index  $i$ .*

# Chapter 3

## Spectral Complexity

In this chapter, we discuss ways we can measure the ‘complexity’ of a Boolean function, which take the form of various definitions relating to the Fourier coefficients of the function, and the shortest way we can describe the function’s output. This discussion has immediate applications to learning theory, where we desire functions which can easily be learned (the simplest functions), and cryptography, where we desire functions which cannot be easily deciphered.

### 3.1 Spectral Concentration

One way for a function to be ‘simple’ is for its Fourier coefficients to be concentrated on coefficients of small degree. Call a Boolean function  $f$   $\varepsilon$ -**concentrated** on degree  $n$  if  $\mathbf{W}^{\geq n}(f) \leq \varepsilon$ . For Boolean valued functions, this is equivalent to saying that  $\mathbf{P}(|S| > n) \leq \varepsilon$ , where  $S$  has the induced spectral distribution. The total influence provides an initial estimate of the spectral concentration.

**Theorem 3.1.** *A Boolean  $f$  is  $\varepsilon$ -concentrated on degree  $n$ , for  $n \geq \mathbf{I}(f)/\varepsilon$ .*

*Proof.* If we write

$$\mathbf{I}(f) = \sum_k k \mathbf{W}^k(f) \leq \varepsilon n$$

then  $n \mathbf{W}^{\geq n}(f) \leq \sum_k k \mathbf{W}^k(f)$ , and combining these inequalities completes the proof. For boolean valued functions, this is essentially Markov’s inequality applied to the spectral distribution.  $\square$

**Example.** The tribes function  $\text{Tribes}_{w,2^w}$  has total influence  $O(\log(n))$ , hence for any constant  $c$ , the tribes function is  $c$ -concentrated on degree up to  $O(\log(n))$ .

Another estimate of the spectral concentration is obtained from the noise stability.

**Theorem 3.2.** For any Boolean-valued function  $f$ , and  $0 < \delta \leq 1/2$ , the Fourier spectrum of  $f$  is  $\varepsilon$ -concentrated on degree  $n \geq 1/\delta$  for

$$\varepsilon = \frac{2}{1 - e^{-2}} \text{NS}_\delta(f) \leq 3 \text{NS}_\delta(f)$$

*Proof.* Using the spectral formula for the noise sensitivity, if  $S$  takes on the spectral distribution, then applying the Markov inequality, we find

$$\begin{aligned} 2\text{NS}_\delta(f) &= \mathbf{E}[1 - (1 - 2\delta)^{|S|}] \\ &\geq (1 - (1 - 2\delta)^{1/\delta}) \mathbf{P}(|S| \geq 1/\delta) \\ &\geq (1 - e^{-2}) \mathbf{P}(|S| \geq n) \end{aligned}$$

where we used that  $1 - (1 - 2\delta)^k$  is a non-negative, non-decreasing function of  $k$ , and  $(1 - 2\delta)^{1/\delta} \leq e^{-2}$ .  $\square$

**Example.** For  $\delta$  sufficiently small, and  $n$  sufficiently large, the noise sensitivity of  $\text{Maj}_n$  with parameter  $\delta$  is bounded by  $\sqrt{\delta}$ . Hence  $\text{Maj}_n$  is  $3\sqrt{\delta}$  concentrated on degree  $m$  for  $m \geq 1/\delta$ .

We can push  $\varepsilon$ -concentration all the way down to 0-concentration on degree  $n$ , which is the same as saying the Fourier expansion of the function in question has degree less than or equal to  $n$ . For these function, we can actually control how many coordinates of the function are useful to the function.

**Lemma 3.3.** If  $f$  is a real-valued Boolean function with  $f \neq 0$ , and the degree of the Fourier expansion is less than or equal to  $m$ , then  $\mathbf{P}(f(X) \neq 0) \geq 2^{-m}$ .

*Proof.* We prove this by induction on the number of coordinates of  $f$ . If  $f(x) = a$  is a constant function, and  $a \neq 0$ , then  $\mathbf{P}(f(X) \neq 0) = 1 \geq 2^{-0}$ . If  $f(x) = a + bx$ , and  $b \neq 0$ , then either  $f(1)$  or  $f(-1)$  are non-zero, so the lemma is satisfied here. In general, if we write

$$f(x) = \sum \hat{f}(S) x^S$$

and then we define  $n-1$  dimensional functions  $f_1(x) = f(x, 1)$ , and  $f_{-1}(x) = f(x, -1)$ , then

$$f_1(x) = \sum_{n \notin S} \left( \hat{f}(S) + \hat{f}(S \cup \{i\}) \right) x^S \quad f_{-1}(x) = \sum_{n \notin S} \left( \hat{f}(S) - \hat{f}(S \cup \{i\}) \right) x^S$$

Then either  $f_1$  has degree  $m-1$ , or  $f_{-1}$  has degree  $m-1$ , and it follows by induction that

$$\mathbf{P}(f(X) \neq 0) = \frac{\mathbf{P}(f_1(X) \neq 0) + \mathbf{P}(f_{-1}(X) \neq 0)}{2} \geq \frac{1}{2^{(m-1)}} \frac{1}{2} = 2^{-m}$$

and by induction, the bound holds for all functions  $f$ .  $\square$

Using this lemma, since  $\text{Inf}_i(f) = \mathbf{P}((D_i f)(X) \neq 0)$ , and if  $\deg(f) \leq k$ , then  $\deg(D_i f) \leq k-1$ , so either  $\text{Inf}_i(f) = 0$ , or  $\text{Inf}_i(f) \geq 2^{1-k}$ .

**Theorem 3.4.** *If  $\deg(f) \leq k$ , then  $f$  is a  $k2^{k-1}$  junta.*

*Proof.* Because  $\text{Inf}_i(f) = 0$  or  $\text{Inf}_i(f) \geq 2^{1-k}$ , the number of coordinates with non-zero influence on  $f$  is at most  $\mathbf{I}(f)/2^{1-k}$ , and those coordinates with non-zero influence are irrelevant to the functions definition. Since

$$\mathbf{I}(f) = \sum_{m \leq k} m \mathbf{W}^m(f) \leq k(\mathbf{W}^1(f) + \dots + \mathbf{W}^k(f)) = k$$

hence  $\mathbf{I}(f)/2^{1-k} \leq k2^{k-1}$ , and so  $f$  is a  $k2^{k-1}$  junta.  $\square$

The Friedgut Kalai Naor theorem is essentially a more robust version of this theorem, in the case where  $k = 1$ . What's more, we shall find that the next section gives us a reason why we can't improve this theorem by much more, because decision trees provide us with a method of constructing  $\deg(f) \leq k$  functions which can access  $k2^{k-1}$  coordinates.

## 3.2 Decision Trees

Another classification of 'simple' Boolean functions occurs by analyzing the decision trees which represent the function. A **decision tree** for a function  $f : \mathbf{F}_2^n \rightarrow \mathbf{R}$  is a representation of  $f$  by a binary tree in which the internal nodes of the tree are labelled by variables, edges labelled 0

or 1, and leaves by real numbers, such that  $f(x)$  can be obtained by starting at the head of the tree, then proceeding down based on the input and the labels of the tree. The **size** of the tree is the number of leaves, and the **depth** is the maximum possible length of a branch from root to leaf. The depth counts the maximum number of questions we need to ask to determine the output of a function  $f$ .

Essentially, a decision tree subdivides  $\mathbf{F}_2^n$  into cubes upon which the represented function is constant. These cubes can be identified as the set of inputs which follows the same path  $P$ , and we shall denote the cube by  $C_P$ . It follows that

$$f(x) = \sum_{\text{paths } P \text{ of } T} f(P) \mathbf{I}(x \in C_P)$$

This essentially tells us that low-depth decision trees should have low spectral complexity, because the length of paths bound the degree of the Fourier coefficients. Introducing terminology which will soon become more important, we define the **sparsity** of a function  $f$ , denoted  $\text{sparsity}(f)$ , to be the number of elements of its domain which are non-zero. We let the **spectral sparsity** of  $f$  be  $\text{sparsity}(\hat{f})$ , it is the number of subsets  $S \subset [n]$  such that  $\hat{f}(S) \neq 0$ . Also define  $f$  to be  $\varepsilon$ -**granular** if  $f(x)$  is a multiple of  $\varepsilon$  for any  $x$  in the domain.

**Theorem 3.5.** *If  $f$  can be computed by a decision tree of size  $s$  and depth  $k$ , then*

- (1)  $\deg(f) \leq k$ .
- (2)  $\text{sparsity}(\hat{f}) \leq s 2^k \leq 4^k$
- (3)  $\|\hat{f}\|_1 \leq s \|f\|_\infty \leq 2^k \|f\|_\infty$ .
- (4)  $\hat{f}$  is  $2^{-k}$ -granular, assuming  $f$  is integer valued.

*Proof.* If a path  $P$  has depth  $m$ , querying if  $x_{i_1} = a_{i_1}, x_{i_2} = a_{i_2}, \dots, x_{i_m} = a_{i_m}$ , then

$$\begin{aligned} \mathbf{I}(x \in C_P) &= \mathbf{I}(x_{i_1} = a_{i_1}, \dots, x_{i_m} = a_{i_m}) \\ &= \frac{1}{2^m} \prod_{j=1}^m (1 + a_j x_{i_j}) = \frac{1}{2^m} \sum_{S \subset \{i_1, \dots, i_m\}} a^S x^S \end{aligned}$$



which we see has a Fourier expansion with degree less than or equal to  $m$ . It follows that if each path determining  $f$  has degree bounded by  $k$ , then the degree of the Fourier coefficients of  $f$  are bounded by  $k$ , because of the sum formula we calculated above, hence (1) follows. We also see that each path of length  $m$  adds at most  $2^m$  Fourier coefficients to the equation, hence (2) follows (also, we see that similar branches share many of the same Fourier coefficients, hence this bound can often be tightened). What's more,

$$f(x) = \sum_{\text{paths } P \text{ of } T} \frac{f(P)}{2^{l(P)}} \sum_{S \subset \{i_1^P, \dots, i_{l(P)}^P\}} a_P^S x^S$$

Thus

$$\|\hat{f}\|_1 \leq \sum_{\text{paths } P \text{ of } T} \sum_{S \subset \{i_1^P, \dots, i_{l(P)}^P\}} \frac{|f(P)|}{2^{l(P)}} \leq s \|f\|_\infty$$

hence (3) follows. Since  $a_P^S \in \{-1, 1\}$ , if  $f(P) \in \mathbf{Z}$  then  $f(P)a_P^S/2^{l(P)}$  is  $2^{-k}$  granular, hence, summing up, we find all the coefficients of  $f$  are.  $\square$

**Theorem 3.6.** *If  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  is computable by a decision tree of size  $s$ , and  $0 < \varepsilon \leq 1$ , then the spectrum of  $f$  is  $\varepsilon$  concentrated on degree  $m \geq \log(s/\varepsilon)$ .*

*Proof.* Let  $T$  be the decision tree of size  $s$ , and consider the tree  $T_0$  obtained by truncating each branch of length greater than or equal to  $\log(s/\varepsilon)$ . New leaves created can be chosen arbitrarily. Then the function  $g$  obtained from  $T_0$  is  $\varepsilon$ -close to  $f$ , because  $g$  only differs from  $f$  on the paths of length  $l \geq \log(s/\varepsilon)$ , and since each query the decision tree makes subdivides the domain in two, changes only  $2^{-l} \leq 2^{-\log(s/\varepsilon)} = \varepsilon/s$  elements of the domain, and since we only have  $s$  paths, we have only changed  $\varepsilon$  elements of the domain. Since if  $m \geq \log(s/\varepsilon)$ ,  $\hat{g}(S)$  vanishes for  $|S| \geq m$ , hence

$$\mathbf{W}^{\geq m}(f) \leq \|\hat{g} - \hat{f}\|_2^2 = \|g - f\|_2^2 \leq 4\varepsilon$$

COME UP WITH TIGHTER BOUND LATER.  $\square$

### 3.3 Computational Learning Theory

Computational learning theory attempts to solve the following problem, motivated from statistics. Given a class of functions  $\mathcal{C}$ , we attempt to determine an unknown **target function**  $f \in \mathcal{C}$  with limited access to that

function. We shall assume, of course, that our Boolean-valued functions. The two models of obtaining data to learn the target function are:

- **random examples**, where we draw random samples  $(X, f(X))$ , where  $X$  is uniformly random on the domain.
- **queries**, where we can request a specific value  $x$  for any  $x \in \{-1, 1\}^n$ .

The query method is at least as powerful as the random example method, because we can always randomly choose  $x$  ourselves. After a certain number of steps, we are required to choose a **hypothesis function**  $h \in \mathcal{C}$ . We say that an algorithm **learns**  $\mathcal{C}$  **with error**  $\varepsilon$  if for any  $f \in \mathcal{C}$ , the algorithm outputs a hypothesis function (not necessarily in  $\mathcal{C}$ ) which is  $\varepsilon$ -close to  $f$  with high probability. The particular accuracy probability baseline is irrelevant, since we can always adjust the baseline by introducing a polynomial number of steps.

As should be intuitive, the more ‘simple’ the class  $\mathcal{C}$  is, the faster it should be to recognize a hypothesis function. If  $\mathcal{C}$  is suitably complex, then exponential running time may be necessary. Generalizing our definition of concentration of spectrum, we say  $f$  is  $\varepsilon$  concentrated on  $\mathcal{F} \subset 2^{[n]}$  if

$$\sum_{S \notin \mathcal{F}} \hat{f}(S)^2 \leq \varepsilon$$

The main result is that learning a function is equivalent to learning its Walsh Fourier spectrum, so that functions concentrated on a small portion of the Fourier space are simpler to learn.

**Theorem 3.7.** *Given access to random samples to a Boolean-valued function, there is a randomized algorithm which take  $S \subset [n]$  as input,  $0 \leq \delta, \varepsilon \leq 1/2$ , and outputs an estimate  $\tilde{f}(S)$  for  $\hat{f}(S)$  such that*

$$\mathbf{P}(|\tilde{f}(S) - \hat{f}(S)| > \varepsilon) \leq \delta$$

*in  $\log(1/\delta) \cdot \text{Poly}(n, 1/\varepsilon)$  time.*

*Proof.* We have  $\hat{f}(S) = \mathbf{E}[f(X)X^S]$ . Given random samples

$$(X_1, f(X_1)), \dots, (X_m, f(X_m))$$

we can compute  $f(X_k)X_k^S$ , and then estimate  $\mathbf{E}[f(X)X^S]$ . A standard application of the Chernoff bound shows that  $O(\log(1/\delta)/\varepsilon^2)$  examples are sufficient to obtain an estimate within  $\pm\varepsilon$  with probability at least  $1 - \delta$ .  $\square$

**Theorem 3.8.** *If  $f$  is a Boolean-valued function,  $g$  is a real-valued Boolean function, and  $\|f - g\|_2^2 \leq \varepsilon$ . Let  $h(x) = \text{sgn}(g(x))$ , with  $\text{sgn}(0)$  chosen arbitrarily. Then  $d(f, h) \leq \varepsilon$ .*

*Proof.* Since  $|f(x) - g(x)|^2 \geq 1$  when  $f(x) \neq \text{sgn}(g(x))$ ,

$$d(f, h) = \mathbf{P}(f(X) \neq h(X)) \leq \mathbf{E}[|f(X) - g(X)|^2] = \|f - g\|_2^2$$

Thus we can accurately ‘discretize’ a real-valued estimate of a Boolean-valued function.  $\square$

These theorems allow us to show that estimating Fourier coefficients suffices to estimate the function.

**Theorem 3.9.** *If an algorithm has random sample access to a Boolean function  $f$ , and can identify a family  $\mathcal{F}$  upon which  $f$ ’s Fourier spectrum is  $\varepsilon$  concentrated, then in  $\text{poly}(|\mathcal{F}|, n, 1/c)$  time, we can with high probability produce a hypothesis function  $\varepsilon + c$  close to  $f$ .*

*Proof.* For each  $S \in \mathcal{F}$ , we produce an estimate  $\tilde{f}(S)$  such that

$$|\tilde{f}(S) - \hat{f}(S)| \leq \delta$$

except with probability bounded by  $\rho$ , in  $\log(1/\rho) \cdot \text{poly}(n, 1/\delta)$  time. Applying a union bound, we find all inequalities hold except with probability  $\delta|\mathcal{F}|$ . Finally, we form the function  $g(x) = \sum \tilde{f}(S)x^S$ , and then output  $h(x) = \text{sgn}(g(x))$ . Now by the last lemma, it suffices to bound the  $L_2$  norm of  $f$  and  $g$ , and

$$\begin{aligned} \|f - g\|_2^2 &= \sum \left( \hat{f}(S) - \hat{g}(S) \right)^2 \\ &= \sum_{S \in \mathcal{F}} \left( \hat{f}(S) - \tilde{f}(S) \right)^2 + \sum_{S \notin \mathcal{F}} \hat{f}(S)^2 \\ &= |\mathcal{F}|\delta^2 + \varepsilon \end{aligned}$$

Hence  $h$  is  $|\mathcal{F}|\delta^2 + \varepsilon$  close to  $f$ , and we have computed  $h$  in  $\log(1/\rho) \cdot \text{poly}(n, 1/\delta)$  time. If we let  $\delta = \sqrt{c/|\mathcal{F}|}$ , we find  $h$  is  $c + \varepsilon$  close to  $f$ , in time  $\log(1/\rho) \cdot \text{poly}(n, |\mathcal{F}|, 1/c)$ .  $\square$

If we assume that we are learning over a simple concept class to begin with, this theorem essentially provides all the information we need to know in order to learn efficiently over this concept class.

**Example.** If our concept class consists of elements in  $\mathcal{C}$ , which only contains functions of degree  $\leq k$ , then  $\mathcal{C}$  is 0-concentrated on the Fourier coefficients of degree  $k$ , which contains

$$|\mathcal{F}| \leq \sum_{i=0}^k \binom{n}{i} = O(n^k)$$

elements. Thus in  $\log(1/\rho) \cdot \text{poly}(n^k, 1/\varepsilon)$ , we can learn an element of  $\mathcal{C}$  with probability  $1 - 1/\rho$  up to an accuracy  $\varepsilon$ .

**Example.** If  $\mathcal{C}$  consists of elements  $f$  with  $\mathbf{I}(f) \leq t$ , then each element is  $\varepsilon$  concentrated on degree past  $t/\varepsilon$ , thus we can learn elements of this set in  $\log(1/\rho) \cdot \text{poly}(n^{t/\varepsilon}, 1/c)$  with error probability  $\rho$ , and error  $\varepsilon + c$ .

**Example.** If  $\mathcal{C}$  consists solely of monotone functions, then the total influence of the functions is bounded by  $\sqrt{2n/\pi} + O(1/\sqrt{n}) = O(\sqrt{n})$ , hence we can learn elements of  $\mathcal{C}$  in  $\log(1/\rho) \cdot \text{poly}(n^{O(\sqrt{n})/\varepsilon}, 1/c)$  with error probability  $\rho$ , and error  $\varepsilon + c$ .

**Example.** If a class  $\mathcal{C}$  consists of functions  $f$  such that  $\mathbf{NS}_\delta(f) \leq 3\varepsilon$ , for  $\delta \in (0, 1/2]$ , then  $f$  is  $\varepsilon$  concentrated on degree  $1/\delta$ , hence  $\mathcal{C}$  is learnable with error  $\varepsilon + c$  and error probability  $\rho$  in time  $\log(1/\rho) \cdot \text{poly}(n^{1/\delta}, 1/c)$ .

**Example.** If a class  $\mathcal{C}$  consists of functions  $f$  which can be represented by a decision tree with size bounded by  $s$ , then the spectrum of  $f$  is  $\varepsilon$ -concentrated on degree  $\log(s/\varepsilon)$ , hence  $\mathcal{C}$  is learnable with error  $\varepsilon + c$  and error probability  $\rho$  in time  $\log(1/\rho) \cdot \text{poly}(n^{\log(s/\varepsilon)}, 1/c) = \log(1/\rho) \cdot \text{poly}(s/\varepsilon, 1/c)$ .

### 3.4 Goldreich-Levin Theorem

We have now reduced our problem to identifying a family  $\mathcal{F}$  upon which  $f$  is concentrated. One such method is provided by the Goldreich-Levin algorithm, which can find  $\mathcal{F}$  assuming query access to  $f$ . In order for the algorithm to terminate in polynomial time, we require that there is a set

$\mathcal{F}$  which exists and is small, yet the algorithm will find a concentration family  $\mathcal{F}$  given arbitrary input.

First, we consider the algorithm's origin in cryptography. The goal of this field of study is to find 'encryption functions'  $f$  such that if  $x$  is a message, then  $f(x)$  is easy to compute, but it is very difficult to compute  $x$  given  $f(x)$ . The strength of the encryption  $f$  is the difficulty in how it can be inverted. The Goldreich Levin theorem is a tool for building strong encryption schemes from weak schemes. Essentially, this breaks down into finding linear function slightly correlated to some fixed function  $f$ , given query access to the function.

The Goldreich Levin theorem assumes query access to a Boolean-valued function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , and some fixed  $\tau > 0$ . Then in time  $\text{poly}(n, 1/\tau)$ , the algorithm outputs a family  $\mathcal{F}$  of subsets of  $[n]$  such that

1. If  $|\hat{f}(S)| \geq \tau$ ,  $S \in \mathcal{F}$ .
2. If  $S \in \mathcal{F}$ ,  $|\hat{f}(S)| \geq \tau/2$ .

Given a particular  $S$ , we know we can compute the estimated Fourier coefficients  $\tilde{f}(S)$  to an arbitrary degree of precision with a high degree of accuracy. The problem is that if we did this for all  $S$ , then our algorithm wouldn't terminate in polynomial time. The Goldreich-Levin algorithm uses a divide-and-conquer strategy to measure the Fourier weight of the function over various collections of sets.

Given  $f : \{-1, 1\}^n \rightarrow \mathbf{R}$ , it will be more natural to think of the function as  $f : \{-1, 1\}^{[n]} \rightarrow \mathbf{R}$ , i.e. the function which takes an input  $x : [n] \rightarrow \{-1, 1\}$ , and outputs a number  $f(x)$ . We can consider the Fourier expansion of an arbitrary  $f : \{-1, 1\}^A \rightarrow \mathbf{R}$ , where  $A$  is an arbitrary index set, and then the characters will take the form  $x^S$  for  $S \subset A$ , and we can consider the Fourier coefficients. Let  $(J, J^c)$  be a partition of  $[n]$ . Then for  $z \in \{-1, 1\}^J$ , we can define  $f_{J|z} : \{-1, 1\}^{J^c} \rightarrow \mathbf{R}$  to be the function obtained by fixing all indices of  $J$  with the values  $z$ . It follows that if  $S \subset J^c$ , then

$$\hat{f}_{J|z}(S) = \sum_{T \subset J} \hat{f}(S \cup T) z^T$$

Then if we pick  $z$  randomly over all choices, we find

$$\mathbf{E}_Z[\hat{f}_{J|Z}(S)] = \sum_{T \subset J} \hat{f}(S \cup T) \mathbf{E}[Z^T] = \hat{f}(S)$$

If we define  $F_{S|J^c} : \{-1, 1\}^{J^c} \rightarrow \mathbf{R}$ , for  $S \subset J$ , defined by

$$(F_{S|J^c}f)(z) = \widehat{f_{J|Z}}(S) = \sum_{T \subset J^c} \widehat{f}(T \cup S) z^T$$

Then this gives a Fourier expansion for the function  $(F_{S|J^c}f)$  and

$$\mathbf{E}_Z[\widehat{f_{J|Z}}(S)^2] = \mathbf{E}[(F_{S|J^c}f)(Z)^2] = \sum_{T \subset J^c} \widehat{f}(T \cup S)^2$$

Given  $f$  and  $S \subset J \subset [n]$ , let

$$\mathbf{W}^{S|J^c}(f) = \sum_{T \subset J^c} \widehat{f}(S \cup T)^2$$

which is the weight over irrelevant indices. A crucial tool of Goldreich-Levin is that this is the expected value of  $\widehat{f_{J|Z}}(S)^2$ .

**Theorem 3.10.** *For any  $S \subset J \subset [n]$ , an algorithm with query access to  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  can compute an estimate of  $\mathbf{W}^{S|J^c}(f)$  that is accurate to within  $\pm \varepsilon$ , except with probability at most  $\delta$ , in  $\text{poly}(n, 1/\varepsilon) \cdot \log(1/\delta)$  time.*

*Proof.* We can write

$$\begin{aligned} \mathbf{W}^{S|J^c}(f) &= \mathbf{E}_Z[\widehat{f_{J|Z}}(S)^2] = \mathbf{E}_Z \left[ \mathbf{E}_Y \left[ f_{J|Z}(Y) Y^S \right]^2 \right] \\ &= \mathbf{E}_Z \left[ \mathbf{E}_{Y_0, Y_1} \left[ f_{J|Z}(Y_0) Y_0^S f_{J|Z}(Y_1) Y_1^S \right] \right] \end{aligned}$$

using queries to  $f$ , we can sample the inside of this equation, and a Chernoff bound shows that  $O(\log(1/\delta)/\varepsilon^2)$  samples are enough for the estimate to have accuracy  $\varepsilon$  with confidence  $1 - \delta$ .  $\square$

We can now describe the Goldreich Levin algorithm. Initially, all subsets of  $[n]$  are put in a single ‘bucket’. We then repeat the following process:

- Select any bucket  $B$  with  $2^m$  sets in it.
- Split  $B$  into two buckets  $B_1$  and  $B_2$  of  $2^{m-1}$  sets.
- Estimate  $\sum_{U \in B_1} \widehat{f}(U)^2$  and  $\sum_{U \in B_2} \widehat{f}(U)^2$ .

- Discard  $B_1$  and  $B_2$  if the weight estimate is less than or equal to  $\tau^2/2$ .

The algorithm stops when each bucket contains a single item. Note that we never discard a set  $S$  with  $|\hat{f}(S)| \geq \tau$ , because  $\tau^2 \geq \tau^2/2$ . Furthermore, if a bucket containing two elements splits into two buckets containing a single element, then if  $S$  is undiscarded in this process, we must have  $|\hat{f}(S)| \geq \tau/2$ , else  $|\hat{f}(S)|^2 \leq \tau^2/4 \leq \tau^2/2$ . Note that we need only calculate the estimates up to  $\pm\tau^2/4$  for the algorithm to be correct, so we have some wriggle room to work with.

Now we need to determine how fast it takes for the algorithm to terminate. Every bucket that isn't discarded has weight  $\tau^2/4$ , so Parseval's theorem tells us that there is never more than  $4/\tau^2$  active buckets. Each bucket can only be split  $n$  times, hence the algorithm repeats its main loop  $4n/\tau^2$  times. If the buckets can be accurately weighed and maintained in  $\text{poly}(n, 1/\tau)$  time, the overall running time will be  $\text{poly}(n, 1/\tau)$ .

Finally we describe how to weight the buckets. Let

$$B_{k,S} = \{S \cup T : T \subset \{k+1, k+2, \dots, n\}\}$$

Then  $|B_{k,S}| = 2^{n-k}$ , the initial bucket is  $B_{0,\emptyset}$ , and any bucket  $B_{k,S}$  splits into  $B_{k+1,S}$  and  $B_{k+1,S \cup \{k\}}$ . The weight of  $B_{k,S}$  is  $\mathbf{W}^{S \cup \{k+1, \dots, n\}}(f)$ , and can be measured to accuracy  $\pm\tau^2/4$  with confidence  $1 - \delta$  in  $\text{poly}(n, 1/\tau) \cdot \log(1/\delta)$ . The algorithm needs at most  $8n/\tau^2$  weighings, hence by setting  $\delta = \tau^2/(80n)$ , we can ensure all weights are accurate with high probability, and the algorithm is  $\text{poly}(n, 1/\tau)$ .

# Chapter 4

## Property Testing

We now study property testing, probabilistically checkable proofs, and constraint satisfaction problems. All our work is centered around *dictatorship testing* – checking if an arbitrary boolean-valued function is a dictator. It turns out that being able to test if a certain function is a dictator is sufficient to test *any* property of boolean-valued functions, provided that the right ‘proof’ is provided of the property.

The field of property testing asks a simple problem. Given a collection  $\mathcal{C}$  of  $n$ -bit Boolean-valued functions, and a function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , how easy is to determine if  $f \in \mathcal{C}$  using (not necessarily uniformly) random examples or queries of the function. An  **$n$ -query testing algorithm** for  $\mathcal{C}$  is an algorithm which randomly chooses elements  $x_1, \dots, x_n \in \{0, 1\}^n$ , queries  $f(x_1), \dots, f(x_n)$ , and decides deterministically whether  $f \in \mathcal{C}$ . An  **$n$ -query local tester** with rejection rate  $\lambda > 0$  is a test which always accepts  $f$  if  $f \in \mathcal{C}$ , and if  $d(f, \mathcal{C}) > \varepsilon$ , then the tester rejects  $f$  with probability exceeding  $\lambda \varepsilon$ . Taking the converse, we obtain our original property testing statements – if  $f$  is accepted with probability greater than  $1 - \varepsilon$ , then  $d(f, \mathcal{C}) \leq \varepsilon / \lambda$ .

**Example.** *The BLR test is a 3-query local tester with rejection rate 1 for the class of linear functions (which acts like a 1/3 rejection rate for small enough  $\varepsilon$ ). Technically, the BLR test is really a family of testers for each dimension of Boolean function, but we don’t need to be that pedantic.*

We could also consider adaptive testing algorithms, which allow us to choose each  $x_i$  sequentially after viewing  $f(x_1), \dots, f(x_{i-1})$ , or we could



consider local testers with more general rejection rates (quadratic or logarithmic instead of linear, for instance), and to allow the number of queries to depend on the error rate, but we'll focus on the current scenario for now.

## 4.1 Dictator Tests

Recall that a dictator function is a character  $x^i$ , for some  $i \in [n]$ , and we shall denote the set of dictators by  $\mathcal{D}$ . We shall find that the **dictator testing** problem, which asks us to determine whether an arbitrary Boolean-valued function is a dictator, is one of the most important property testing problems there is.

A dictator test could surely start by testing for linearity. Using a union bound, this would reduce the problem to testing whether an arbitrary character  $x^S$  is a dictator. Classically, this was the method for constructing dictator tests. For instance, the dictators are the only characters which satisfy

$$\text{and}(x^S, y^S) = \prod_{i \in S} \text{and}(x_i, y_i)$$

And so we can test whether this equation holds to determine whether we have a dictator. But using Arrow's theorem, and a bit of extra work, we can come up with a much more intelligent test.

Given a Boolean-valued function  $f$ , take three random inputs  $X, Y$ , and  $Z$  uniformly from the 6 possible random triples  $(X, Y, Z)$  such that  $\text{NAE}(X, Y, Z)$  holds, and compute  $\text{NAE}(f(X), f(Y), f(Z))$ . Accept  $f$  if the outputs are not all equal. We have shown that if  $f$  is accepted with probability  $1 - \varepsilon$ , then  $\mathbf{W}^1(f) \geq 1 - (9/2)\varepsilon$ , and the FKN theorem then implies that  $f$  is  $O(\varepsilon)$  close to a dictator or its negation. However, this isn't exactly a dictator test, and requires a very difficult theorem to obtain a guarantee on closeness. Fortunately, if we do a NAE test and a BLR test sequentially, then we obtain a 6 query 0.1 rate dictator test, without using the FKN test.

**Theorem 4.1.** *Suppose that  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  passes both the BLR test and the NAE test with probability  $1 - \varepsilon$ .*

*Proof.* We then know that  $f$  passes the BLR test with at least probability  $1 - \varepsilon$ , hence  $f$  is  $\varepsilon$  close to some linear function  $x^S$ , and  $f$  passes the NAE test with probability  $1 - \varepsilon$ , hence  $\mathbf{W}^1(f) \geq 1 - (9/2)\varepsilon$ . But if  $S$  contains  $k$

elements, then  $\mathbf{W}^k(f) \geq \hat{f}(S)^2 \geq (1 - 2\varepsilon)^2 = 1 - 4\varepsilon + 4\varepsilon^2$ . If  $k \neq 1$ , then  $2 + 4\varepsilon^2 - (4 + 9/2)\varepsilon \leq 1$ , which implies that  $\varepsilon \geq 0.1$ , and therefore  $f$  is certainly  $10\varepsilon$  close to being linear. Otherwise, we find  $f$  is  $\varepsilon$  close to a linear function.  $\square$

We aren't that interested in finding a large rejection rate, but rather finding an algorithm that uses the least queries possible. There is a simple trick to reducing the number of queries of a property test, at the cost of rejection rate, when we perform multiple tests at once. Suppose we have a test consisting of consecutively performing multiple subtests with  $q_1, \dots, q_n$  queries, then if we uniformly randomly choose *exactly one* test to do, then if our new algorithm is accepted with probability  $1 - \varepsilon$ , we can guarantee that every other test is accepted with probability  $1 - n\varepsilon$ , and we only do  $\max(q_1, \dots, q_n)$  queries. For instance, we can modify our 6-query 0.1-rate dictator test to a 3-query 0.05-rate dictator test.

One nice property of dictatorship testing is that we can use a dictatorship test to obtain a 3-query 0.01 property test over any property  $\mathcal{C} \subset \mathcal{D}$ . We perform two tests on an input function  $f$ .

1. We perform a 3-query 0.1 rate dictatorship test on  $f$ .
2. Form the string  $y$  by  $y_i = -1$  if  $x^i \in \mathcal{C}$ , and  $y_i = 1$  otherwise. Then  $y^j = -1$  if and only if  $x^j \in \mathcal{C}$ . We perform local decoding on  $f$  to calculate the value of  $f$  if it was linear.

We only do one of these tests are random, so our test is 3-query. If the test is accepted on  $f$  with probability  $1 - \varepsilon$ , then  $f$  is accepted by the dictatorship test with probability  $1 - 2\varepsilon$ , hence  $f$  is  $20\varepsilon$  close to some dictator  $x^j$ . Then we know that, if we perform local decoding, the chance of correctly calculating  $x^j$  is at least  $1 - 40\varepsilon$ . If  $x^j \notin \mathcal{C}$ , this implies that we reject  $f$  with probability at least  $1 - 40\varepsilon$ , but we know that  $f$  is accepted with probability at least  $1 - 2\varepsilon$ , but then we can sum up and find  $1 \geq 2 - 42\varepsilon$ , hence  $\varepsilon \geq 1/42$ , and we find our test rejects with a rate of 0.02.

## 4.2 Probabilistically Checkable Proofs

We have shown that any subset of dictator functions has a 3 query proof. It turns out that testing over *any property* can be reduced to dictator testing, provided we have an appropriate 'proof'. To see this, we must slightly

generalize our notion of property testing. Rather than just considering families of functions  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  to test over, we will be testing over binary strings of a fixed length – elements of  $\{0, 1\}^N$  for a fixed  $N$ . Rather than evaluating a function at positions, we get to randomly query certain bits of the string, and then choose to accept or reject the string. The rejection rates, query amount, and so on all generalize, because testing the value of a function at a certain point of the domain is equivalent to looking at a single bit of the binary string.

**Example.** Consider testing whether a string is the all zero string. Given an input  $x$ , consider randomly selecting an index  $i$ , and then testing whether  $x_i = 0$ . If  $x$  is accepted with probability  $1 - \varepsilon$ , then  $1 - \varepsilon$  percent of the bits in the string  $x$  are equal to 0, hence  $d(x, 0) = \varepsilon$ , so this is a test with rate 1, and with a single query.

**Example.** What about testing if the values of a string are all the same. The natural choice is two queries – given  $x$ , we pick  $i$  and  $j$  and test whether  $x_i = x_j$ . If  $\min(d(f, 0), d(f, 1)) = \varepsilon$ , then

$$\begin{aligned} \mathbf{P}(f \text{ is accepted}) &= \mathbf{P}(x_i = 0 | x_j = 0) \mathbf{P}(x_j = 0) + \mathbf{P}(x_i = 1 | x_j = 1) \mathbf{P}(x_j = 1) \\ &= d(f, 0)^2 + d(f, 1)^2 \\ &= 1 - 2d(f, 0)[1 - d(f, 0)] \end{aligned}$$

and so the test is accepted with probability  $1 - 2\varepsilon(1 - \varepsilon)$ .

**Example.** Consider the property that a string has an odd number of 1s. Then the only local tester for this property must make all  $n$  queries.

Now suppose that, in addition to  $x \in \{0, 1\}^n$ , you are given some  $\Pi \in \{0, 1\}^m$  which is a ‘proof’ that  $x$  satisfies the required property. Then it might be possible to check both  $x$  and  $\Pi$  at the same time, *without ever having to trust the accuracy of  $\Pi$* , and be able to obtain a higher rate algorithm that  $x$  is accepted, assuming that  $\Pi$  is the correct certificate. This is a **probabilistically checkable proof of proximity**. Specifically, an  $n$ -query, length  $m$  probabilistically checkable proof of proximity system for a property  $\mathcal{C}$  with rejection rate  $\lambda$  is an  $n$ -query test, which takes a bit string  $x$ , and a proof  $\pi \in \{0, 1\}^m$ , such that if  $x \in \mathcal{C}$ , there is a proof  $\Pi$  such that  $(x, \Pi)$  is always accepted, and if  $d(x, \mathcal{C}) > \varepsilon$ , then for *every* proof  $\Pi$ , if the test accepts with probability  $1 - \varepsilon$ , then  $x$  must be  $x/\lambda$  close to an element of  $\mathcal{C}$ . We normally care about reducing  $n$  as much as possible, without regard to  $m$  or  $\lambda$ .

**Example.** The property  $\mathcal{O}$  that a string has an odd number of 1s cannot be checked with few queries without a proof, but we have an easy 3-query PCPP system with proof length  $n - 1$ , and rejection rate 1. We expect the proof string to contain the sums  $\Pi_j = \sum_{i \leq j+1} x_i$  in  $\mathbf{F}_2$ . We randomly check that one of the following equation holds

- $\Pi_1 = x_1 + x_2$ .
- $\Pi_j = \Pi_{j-1} + x_{j+1}$ .
- $\Pi_{n-1} = 1$ .

If  $x \in \mathcal{O}$ , then the test is accepted with probability 1. If  $x \notin \mathcal{O}$ , then  $d(x, \mathcal{O}) = 1/n$ , and at least one of these equations fails, hence the probability that  $x$  is rejected is at least  $1/n$ .

**Theorem 4.2.** Any property over length  $n$  bit strings has a 3-query PCPP tester with length- $2^{2^n}$  proofs and rejection rate 0.001.

*Proof.* Let  $\mathcal{C}$  be the required property to test, and consider a bijection  $\pi : [N] \rightarrow \{0, 1\}^n$ , where  $N = 2^n$ . Then each element of  $\{0, 1\}^n$  corresponds to a dictator on  $[N]$ . Given an input  $x$ , we will interpret a proof as a function  $\Pi : \{0, 1\}^N \rightarrow \{0, 1\}$  as a dictator corresponding to  $x$ . The test then becomes a dictator test, which we can already solve with three queries. Thus we consider two separate tests.

- Testing if  $\Pi$  is a dictator, corresponding to an element of  $\pi^{-1}(\mathcal{C})$
- Assuming  $\Pi$  is a dictator corresponding to an element  $y \in \{0, 1\}^n$ , testing whether  $y = x$ .

It is clear that if  $\Pi$  is a correct proof, and  $x \in \mathcal{C}$ , then the test will always be accepted. Otherwise, if  $(x, \Pi)$  is accepted with probability  $1 - \varepsilon$ , then we conclude from the first test that  $\Pi$  is  $200\varepsilon$  close to some dictator  $\chi_\alpha$ , which is an element of  $\pi^{-1}(\mathcal{C})$ .

Now we require a clever way of determining which element of  $\{0, 1\}^n$   $\Pi$  corresponds to, given that  $\Pi$  is a dictator. Define  $x^i \in \{0, 1\}^N$  by  $x_j^i = \pi(i)_j$ . If  $\Pi$  is a correct proof, then  $\Pi(x^i) = \chi_\alpha(x^i) \dots$  finish later.  $\square$

The  $2^{2^n}$  is a pretty bad estimate, but it is essentially required to be able to perform this test on all  $2^{2^n}$  different properties on  $\{0, 1\}^n$ . It should be

reasonable to be able to identify better systems for ‘explicit’ properties. This can be qualified by the complexity of the circuit which can compute the indicator function for the property.

A **PCPP reduction** is an algorithm which takes a property as input, and outputs a PCPP system testing that property. The notions of proof length, query size, etc all transfer to the PCPP reduction. Essentially, a PCPP reduction gives a constructive proof limiting the ability to test general properties. We have proved that there exists a 3-query  $2^{2^n}$  proof length PCPP reduction with rejection rate 0.001. With a little work, we can improve this reduction to have proof length  $2^{\text{poly}(C)}$ , where the size of a property  $C$  is the size of the circuit representing it. There is a much deeper and more dramatic improvement to this property.

**Theorem 4.3 (PCPP).** *There is a 3-query PCPP reduction with proof length polynomial in the size of  $C$ .*

Though the  $2^{2^n}$  PCPP reduction seems unnecessary after we have proved the PCPP theorem, the  $2^{2^n}$  reduction is integral to the proof of the PCPP theorem, and is therefore important even if you want to understand the proof of the PCPP theorem.

### 4.3 Constraint Satisfaction and Property Testing

Recall that an  $m$  arity  $n$  constraint **satisfaction problem** over a set  $\Omega$  is defined by a finite set of predicates  $f_1, \dots, f_n : \Omega^m \rightarrow \{0, 1\}$ . The goal is to find the elements of  $\Omega^m$  which satisfy the most predicates. This is obviously NP complete, since the problem encompasses the SAT problem, so we instead try and find elements of  $\Omega^m$  which satisfy the most predicates, or near to the most elements. We assume each  $f_i$  is an  $s$  junta, for some  $s$ .

**Example.** *The 3-SAT problem on  $m$  variables is an  $m$  arity constraint satisfaction problem, where each constraint is a 3 junta.*

**Example.** *The MAX-CUT problem on a graph with  $n$  nodes and  $m$  edges is to find a partition of the graph so that the most possible edges cross the partitions. A partition corresponds to an element of assignment  $\{0, 1\}^n$ , and this partition is evaluated against the not equal constraints for any edge  $(v, w)$ .*

**Example.** Given a system of linear equations in  $\mathbf{F}_2$ , each containing exactly 3 variables, the MAX-E3-LIN problem asks to find an assignment satisfying as many of the constraints as possible.

String testing and constraint satisfaction are the same problem. Given a constraint instance over  $\Omega^m$  with constraints  $f_1, \dots, f_n$ , and given a fixed solution  $x \in \Omega^m$ , consider the algorithm which randomly choosing  $i \in [n]$  and accepts if  $f_i(x) = 1$ . The probability the algorithm accepts  $x$  is  $\text{val}(f, x)$ . Conversely, if we have a string testing problem on  $\Omega^n$ , with some randomized testing problem, then if we enumerate all random bits used in the algorithm, then each selection of random bits corresponds to a constraint where an element of  $\Omega^n$  is tested, and the value of this constraint satisfaction problem gives the probability that the string testing algorithm is accepted.

## 4.4 Hastad's Hardness Theorem

The PCP theorem indicates that unless  $P = NP$ , it is impossible to approximate 3SAT to arbitrarily close constants. But we can find approximation algorithms for SAT for some constants. For instance, if each clause of MAX-E3SAT contains exactly 3 clauses (we call this subproblem MAX-E3SAT), then each clause is satisfied with probability  $7/8$ , hence in expectation a random assignment will satisfy  $7/8$  of the clauses. There is a way to derandomize this algorithm to obtain a deterministic  $7/8$  approximation algorithm. Though the approximation is trivial, Hastad's theorem says this is the best approximation possible.

**Theorem 4.4** (Hastad). *For any  $\delta > 0$ , if there is an algorithm that returns a 3SAT assignment satisfying at least  $7/8 + \delta$  clauses for MAX-E3SAT for all instances of the problem where it is possible to satisfy all clauses, then  $P = NP$ .*

He also proved a similar optimal hardness of approximation for MAX-E3LIN, a result we will concentrate on for the rest of this section.

**Theorem 4.5** (Hastad). *If there is  $\delta > 0$ , and an approximation algorithm for MAX-E3LIN such that if  $1 - \delta$  of the equations can be satisfied, then the algorithm returns an instance satisfying  $1/2 + \delta$  of the equations, then  $P = NP$ .*

The idea of this theorem is that we can formalize a dictatorship test with a suitably high rejection rate by a reduction to an instance of MAX-E3LIN, and if this reduction was good enough then we would find that  $P = NP$ .

We require some formalization to prove this theorem. Let  $f_1, \dots, f_n$  be predicates over  $\{-1, 1\}$ . If  $0 < \alpha < \beta \leq 1$ , let  $\lambda : [0, 1] \rightarrow [0, 1]$  satisfy  $\lambda(\varepsilon) \rightarrow 0$  as  $\varepsilon \rightarrow 0$ . Suppose that for each  $n$  there is a local tester for functions  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  such that

- If  $f$  is a dictator, then the test accepts with probability at least  $\beta$ .
- If  $\text{Inf}_i^{1-\varepsilon}(f) \leq \varepsilon$ , then the test accepts with probability at most  $\alpha + \lambda(\varepsilon)$ .
- The tester can be viewed as an instance of MAX-CSP( $f$ ).

Then we call this family an  $(\alpha, \beta)$  Dictator vs. No Notables Test using the predicates  $f$ . It is essential that  $\lambda = o_\varepsilon(1)$ , at a rate independent of  $n$ , but we care very little about the rate of convergence to zero.

There is a Blackbox result which relates Dictator vs. No Notables to hardness of approximation results.

**Theorem 4.6.** *Fix a CSP over the domain  $\{-1, 1\}$  with predicates  $f_1, \dots, f_n$ . Given an  $(\alpha, \beta)$  dictator vs. no notables test using predicates  $f_i$ , then for all  $\delta > 0$ , either the universal game conjecture holds, or there is a  $(\alpha + \delta, \beta - \delta)$  approximation for the CSP.*

We rely on the following theorems, beyond our scope, to prove Hastad's theorem.

**Theorem 4.7.** *For any  $0 < \delta < 1/8$ , there is a  $(7/8 + \delta, 1)$  dictator vs no notables test which uses logical or functions on three literals as its predicates.*

**Theorem 4.8.** *For any  $0 < \delta < 1/2$ , there is a  $(1/2, 1 - \delta)$  dictator vs no notables test using 3 variable  $\mathbf{F}_2$  linear equations as its predicates.*

The theorems, combined with the blackbox result above, imply a weak version of Hastad's uniqueness conjecture, in the sense that if there is an approximation, then the unique games conjecture holds, rather than  $P = NP$ .

We cannot use the BLR test to find a  $(1/2, 1 - \delta)$  dictator vs no notables test using 3 variable  $F_2$  linear equations, because it doesn't accept functions with no notable coordinates with probability close to  $1/2$ . The two problems are the constant 0 function and large parity functions. This is fixed by the odd BLR test. Given query access to  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ , we pick  $X, Y \in \{-1, 1\}^n$  randomly, find  $z \in \{-1, 1\}$  randomly, and set  $Z = XY(z, \dots, z)$ . We then accept the test if  $f(X)f(Y)f(Z) = z$ . It is easy to show that

**Theorem 4.9.** *The odd BLR test accepts  $f$  with probability*

$$\frac{1}{2} + \frac{1}{2} \sum_{|S| \text{ odd}} \hat{f}(S)^3 \leq \frac{1}{2} + \frac{1}{2} \max_{|S| \text{ odd}} \hat{f}(S)$$

This means that the constant 1 function is ruled out. What remains is to deal with large parity functions. The trick to this is to add noise to the test. Given a small  $\delta > 0$ , we replace  $Z$  with  $Z' \sim N_{1-\delta}(Z)$ . If  $f$  is a dictator, then there is only a  $\delta/2$  chance this will affect the test. On the other hand, if  $f$  is  $x^S$ , where  $S$  is large, there is a high chance this will disturb the chance of passing the test.

Thus Hastad's  $\delta$  test takes  $X, Y$ , and  $z$ , sets  $Z = b(XY)$ , chooses  $Z' \sim N_{1-\delta}(Z)$ , and accept if  $f(X)f(Y)f(Z') = z$ . This is a  $(1/2, 1 - \delta/2)$  dictator vs. no notables test.



# Bibliography

- [1] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, M. Sudan. *Linearity Testing in Characteristic Two*.