Theoretician's Toolkit **Spring 2019**

Lecture 10: Boolean Analysis and Property Testing

Scribes: Debayan Bandyopadhyay, Kobe Wang, and Alex Yu 5/6/2019

This week, we will discuss the analysis of Boolean functions and its application to property testing.

10.1 Introduction

10.1.1 Background and Review

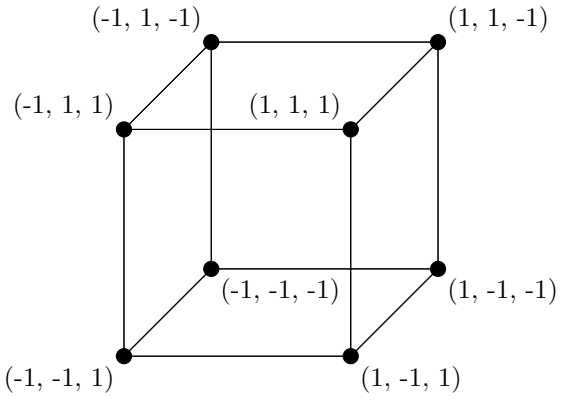
Throughout this lecture, we will be working with Boolean functions, which are functions that map each binary string to one of two values. Note that we will use $\{-1, 1\}$ as opposed to the perhaps more familiar $\{0, 1\}$ for convenience.

Definition 10.1. *(Boolean Function) A Boolean function f is a mapping:*

$$f : \{-1, 1\}^n \to \{-1, 1\} \tag{10.1}$$

Where 1 *represents* False *and* -1 *represents* True.

As you may have seen in classes such as CS 70, the binary strings may be represented as vertices of a hypercube, where two vertices are connected by an edge if their strings differ in only one position. For instance, when $n = 3$ we have the standard cube:



10.1.2 Motivation: Property Testing, Arrow's Theorem

Without testing all possible inputs, we might want to know if a given Boolean function satisfies a certain property, or in other words if f is in P. There are many such examples of properties that we might care about. For example, we might ask if the function is constant, meaning that it returns the same output for every input string, or whether the function is balanced, meaning that it returns 1 for half of the inputs and -1 for the other half of the inputs. A particularly interesting example is testing whether a function is linear, which means that $f(x)f(y) = f(x \cdot y)$ for all possible input strings, x and y. To do this with certainty, we would have to test all possible input strings. But what if we just want to get an answer with a low number of queries which is still close to correct with high probability? Boolean function analysis gives us the tools to tackle this problem.

Interestingly, Boolean function analysis is also a useful tool for a wider range of tasks. Perhaps surprisingly, the technique yields an alternate proof of Arrow's Impossibility Theorem, a key theorem from social choice theory which stipulates that the only election system which satisfies the seemingly simple requirements of *rationality* and *consistency* is a dictatorship.

10.2 Fourier Analysis

Definition 10.2. *(Parity Function) A parity function $\mathcal{X}_S(x)$ for some subset $S \subseteq [n]$ is the Boolean function:*

$$\mathcal{X}_S(x) = \prod_{t \in S} x_i$$

Since each x_i is -1 or 1, this is equivalent to saying that $\mathcal{X}_S(x)$ is 1 if there are an even number of -1's (TRUE) among the subset of variables $\{x_i : i \in S\}$ and -1 else. This is analogous to the familiar parity function

$$\sum_{i \in S} x_i \pmod 2$$

Used when each variable takes a value in $\{0, 1\}$.

Definition 10.3. *(Fourier Transform of Boolean Function) The Fourier transform or Fourier expansion of a Boolean function f is given by*

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\mathcal{X}_S(x)$$

Where $\hat{f}(S)$ is a constant we will call the S-Fourier coefficient of f, and $[n]$ denotes $\{1, \ldots, n\}$.

Observe that this Fourier transform is really just expressing $f(x)$ as a polynomial of the variables x_i, since each $\mathcal{X}_S(x)$ is of the form $x_{i_1}x_{i_2}\cdots x_{i_k}$. Then $\hat{f}(S)$ are the coefficients in front each term of the polynomial. We call this the Fourier transform because it is precisely Hadamard transform for f, which as it turns out is the Fourier transform in the group $\{-1, 1\}^n$ under elementwise multiplication; the Hadamard transform may be seen as a multidimensional size-2 DFT.

Practically, we can calculate the Fourier coefficients using Lagrange interpolation.

Example 10.4. We demonstrate the Fourier expansion of the majority function $n = 3$:

$$\text{MAJ}_3(x) = \begin{cases} 1 & \text{at least half of } x_i \text{ are 1} \\ -1 & \text{else} \end{cases}$$

10.3 Spectral Graph Theory on the Hypercube

Previously we discussed how the input space to any boolean function can be visualized as an n-dimensional hypercube. In this framework, instead of thinking of our functions as mapping from bitstrings to a boolean value, we can instead thinking of our functions as mapping from the vertices of the hypercube to a boolean. In other words, $f : V \to \{-1, 1\}$.

With this framework in mind, we can do spectral graph theory on this hypercube. In spectral graph theory, we usually assign an initial distribution π across the vertices of the graph. Note that the hypercube is symmetric across all its vertices, so it makes the most sense to assign a uniform distribution, $\frac{1}{2^n}$, over all vertices. In terms of the function inputs, this means that the probability distribution over the inputs to the function is uniform.

We also observe that the Boolean functions form a real inner product space, under the following inner product:

Definition 10.5. *(Boolean Function Inner Product) We have the following inner product for Boolean functions $f, g : \{-1, 1\}^n \to \{-1, 1\}$:*

$$\langle f, g \rangle = \mathbb{E}_x[f(x)g(x)] = \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)g(x) \tag{10.2}$$

This dot product measures the *closeness* of f and g. Why is this the case? Consider the term $f(x)g(x)$ for a given vertex (input) x. If $f(x) = g(x)$, then $f(x)g(x) = 1$. If they are not equal, $f(x)g(x) = -1$. Thus, we can express $f(x)g(x)$ as $1 - 2(\mathbb{1}(f(x) \neq g(x)))$, where we get back the same values in the two cases. Thus,

$$\begin{aligned}
\langle f, g \rangle &= \mathbb{E}_x[f(x)g(x)] \\
&= \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)g(x) \\
&= \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} 1 - 2(\mathbb{1}(f(x) \neq g(x))) \\
&= 1 - 2 \sum_{x \in \{-1,1\}^n} \frac{1}{2^n} \mathbb{1}(f(x) \neq g(x)) \\
&= 1 - 2 \sum_{x \in \{-1,1\}^n} \mathbb{P}(x)\mathbb{P}(f(x) \neq g(x) \,|\, x) \\
&= 1 - 2 \sum_{x \in \{-1,1\}^n} \mathbb{P}(x \wedge f(x) \neq g(x)) \\
&= 1 - 2\mathbb{P}_x(f(x) \neq g(x))
\end{aligned}$$

Note that the inner product of f with itself, i.e. $\langle f, f \rangle$, gives a value of 1 as the second term will just be zero.

We let K be the normalized adjacency matrix of the hypercube. For an n-dimensional hypercube (where every vertex has n neighbors), this is just $\frac{1}{n}$ times the adjacency matrix. How can we think about the effect of K on a function f? In fact, we can say that $Kf(x) = \mathop{\mathbb{E}}_{y \text{ neighbor of } x}[f(y)]$. Why is this the case? Suppose we represent f as a column vector with entries $f(x_{(i)})$ where (i) denotes one of the (ordered) 2^n inputs. Then the effect of multiplying the matrix K by the column vector representation of f is to produce a new column vector where the entries of the vector is $\frac{1}{n}$ times the function value of one of the neighbors summed over all of the neighbors of that input vertex. If we think of the $\frac{1}{n}$ factor as the probability that we choose a neighbor of each vertex uniformly at random, this means we have a vector where the entries are $\mathop{\mathbb{E}}_{y \text{ neighbor of } x_{(i)}}[f(y)]$.

Theorem 10.6. *The parity functions $\mathcal{X}_S$ of the subsets of variables are precisely the eigenfunctions of K.*

Proof. We want to show that for every subset S, $KX_S = \kappa_S X_S$ for some κ_S. Let's consider one particular subset, S, and its parity X_S. Suppose $X_S = b$, where $b \in \{-1, 1\}$. What is the expected value of the neighboring vertices of this vertex?

Note that for a hypercube, every neighbor has exactly one bit flipped while the rest of the bits remain the same. This means that for this particular input, there are exactly $n - |S|$ neighboring inputs which have a bit flipped which is not in the subset, and exactly $|S|$ neighboring inputs which have a bit flipped which is in the subset. In the first case, the value of the parity function doesn't change at all, so it stays as b. In the latter case, the parity is $-b$ because exactly one of the values is flipped (i.e. the negative of the bit in the original input).

Then we can directly input that into our expectation formula to get what we want.

$$\begin{aligned}
KX_S &= \mathop{\mathbb{E}}_{y \text{ neighbor of } x}[f(y)] \\
&= \mathbb{E}_{y \text{ neighbor of } x} \frac{1}{n} f(y) \\
&= \frac{1}{n} \mathbb{E}_{y \text{ neighbor of } x} f(y) \\
&= \frac{1}{n}[(n - |S|)b + (|S|)(-b)] \\
&= \frac{1}{n}(n - 2|S|)b \\
&= \frac{n - 2|S|}{n} b \\
&= \big(1 - \frac{2|S|}{n}\big) X_S
\end{aligned}$$

In other words, $KX_S = \kappa_S X_S$ for $\kappa_S = 1 - \frac{2|S|}{n}$, which allows us to conclude that every parity function is an eigenfunction. □

Theorem 10.7. *The parity functions $\mathcal{X}_S$ are orthonormal.*

Proof. What we would like to show is that $\langle X_S, X_T \rangle$ is 0 if $S \neq T$ and 1 if $S = T$. We use the definition of

the inner product to proceed.

$$\begin{aligned}\langle X_S, X_T\rangle &= \mathbb{E}_{x\in\{-1,1\}^n}[X_S X_T] \\ &= \mathbb{E}_x\Big[\prod_{i\in S} x_i \prod_{j\in T} x_j\Big] \\ &= \mathbb{E}_x\Big[\prod_{i\in S\cap T} x_i^2 \prod_{j\in S\triangle T} x_j\Big]\end{aligned}$$

Here, the triangle $\triangle$ represents the symmetric difference. Note that the square of any bit value will just be 1, which gives us

$$\begin{aligned}\langle X_S, X_T\rangle &= \mathbb{E}_x\Big[\prod_{i\in S\cap T} x_i^2 \prod_{j\in S\triangle T} x_j\Big] \\ &= \mathbb{E}_x\Big[\prod_{i\in S\cap T} 1 \prod_{j\in S\triangle T} x_j\Big] \\ &= \mathbb{E}_x\Big[1 \prod_{j\in S\triangle T} x_j\Big]\end{aligned}$$

Now we must proceed by cases. Suppose $S = T$. Then this product of symmetric differences disappears and we just get left with $\mathbb{E}_x(1)$, which of course, is just 1.

Alternatively, if $S \neq T$, then we utilize the fact that every bit is independent (none of the input bits have anything to do with each other) to separate the expectation of the product into a product of expectations.

$$\begin{aligned}\langle X_S, X_T\rangle &= \mathbb{E}_x\Big[\prod_{j\in S\triangle T} x_j\Big] \\ &= \prod_{j\in S\triangle T} \mathbb{E}_x(x_j) \\ &= \prod_{j\in S\triangle T} 0 = 0\end{aligned}$$

Because each bit is equally likely to be a -1 or a 1, the expectation of any bit is just zero. This proves orthonormality. □

Theorem 10.8. *For all $S \subseteq [n]$, $\hat{f}(S) = \langle f, X_S\rangle$. In other words, we can retrieve our Fourier coefficients by simply taking the inner products with our subset parity functions.*

Proof.

$$\begin{aligned}\langle f, X_S\rangle &= \sum_{T\subseteq[n]} \langle \hat{f}(T) X_T, X_S\rangle \\ &= \sum_{T\subseteq[n]} \hat{f}(T)\langle X_T, X_S\rangle\end{aligned}$$

But the inner product is 0 if $S \neq T$ and 1 if $S = T$, so only the term corresponding to subset S survives the summation.

$$\langle f, X_S\rangle = \hat{f}(S)$$

□

Theorem 10.9. *(Parseval's Theorem for Boolean Functions) For Boolean functions f, g we have*

$$\langle f, g\rangle = \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S) \tag{10.3}$$

Proof.

$$\begin{aligned}\langle f, g\rangle &= \mathbb{E}_x[f(x)g(x)] \\ &= \mathbb{E}_x\Big[\sum_{S\subseteq[n]}\sum_{T\subseteq[n]}\hat{f}(S)\hat{g}(T)X_T X_S)\Big]\end{aligned}$$

The only terms surviving the summation are those where $S = T$, so one of the summations disappears.

$$\begin{aligned}\langle f, g\rangle &= \mathbb{E}_x\Big[\sum_{S\subseteq[n]}\hat{f}(S)\hat{g}(S)\Big] \\ &= \sum_{S\subseteq[n]}\hat{f}(S)\hat{g}(S)\end{aligned}$$

□

Earlier we found that $\langle f, f\rangle = 1$. Thus, a corollary is that $\langle f, f\rangle = \sum_{S\subseteq[n]}\big(\hat{f}(S)\big)^2 = 1$.

10.4 Application: Property Testing

Suppose for some Boolean function f, we are provided with a method to query $f(x)$ for any $x \in \{-1, 1\}^n$. We will consider the problem of determining if $f \in \mathcal{P}$ for a given property $\mathcal{P}$. Interestingly, this problem of Boolean property testing is in fact a very general notion. For example, property testing on graphs can be seen as a specific case of Boolean property testing, where $n = \binom{|V|}{2}$ and each f corresponds to a graph $G = (V, E)$ so that $f(e_{uv}) = -1$ iff $uv \in E$, where e_{uv} has -1 only at the index corresponding uv.

Of course we could ask for all 2^n input strings possible to determine if f has the property, but our goal is to minimize the number of queries required, while retaining high probability of correctness. Let us attempt to formalize this notion of testability, in the framework of randomized algorithms you have seen in previous lectures:

Definition 10.10. *(Testable, version 1) A property $\mathcal{P}$ is testable in T queries if there exists an algorithm* ALG *that makes at most T queries for $f(x)$ and satisfies:*

1. *Completeness. If $f \in \mathcal{P}$ then* ALG *outputs* YES *with probability at least $\frac{2}{3}$.*
2. *Soundness. If $f \notin \mathcal{P}$ then* ALG *outputs* NO *with probability at least $\frac{2}{3}$.*

Note: $\frac{2}{3}$ is arbitrary and can be any number $> \frac{1}{2}$. This seems like a pretty good definition, and if an algorithm satisfies this we can run it a polynomial number of times to achieve $1 - \epsilon$ success probability. But actually we have a problem: even for simple properties like $\mathcal{P} = \{f : \forall_x f(x) = 1\}$, i.e. checking if a function is 1 everywhere, for a worst-case function δ defined so that $\delta(x) = 1$ for all but one input, any randomized algorithm has very low probability of finding the particular x and would fail to satisfy soundness. Then let us relax the soundness condition a bit by establishing a notion of how "far" a function is from having a property:

Definition 10.11. *(ϵ-far) We say a function f is ϵ-far from satisfying a property $\mathcal{P}$ iff:*

$$\forall g \in \mathcal{P},\ \Pr_x(f(x) \neq g(x)) > \epsilon$$

Note that for example δ as we defined above would not be ϵ-far from $\mathcal{P}$ for any (universal) constant ϵ. Here is our revised definition of testability:

Definition 10.12. *(Testable, revised) A property $\mathcal{P}$ is ϵ-testable in T queries if there exists an algorithm* ALG *that makes at most T queries for $f(x)$ and satisfies:*

1. *Completeness. If $f \in \mathcal{P}$ then* ALG *outputs* YES *with probability at least $\frac{2}{3}$.*
2. *Soundness. If $f \notin \mathcal{P}$ and f is ϵ-far from $\mathcal{P}$, then* ALG *outputs* NO *with probability at least $\frac{2}{3}$.*

Note that if $f \notin \mathcal{P}$ but is not ϵ-far from $\mathcal{P}$, we make no guarantees. It is possible to consider a harder problem where instead of answering YES or NO we want to approximate the distance to the nearest problem, but for simplicity we will not discuss this. For illustration, we will proceed to prove testability for a simple property.

Example 10.13. $\mathcal{P} = \{f : \forall x,\, f(x) = 1\}$ is ϵ-testable in $O\left(\frac{1}{\epsilon}\right)$ queries.

Proof. The algorithm is as follows: repeat $\frac{2}{\epsilon}$ times: Pick an x uniformly at random, and if $f(x) = -1$ return FALSE and halt. After all queries, if $f(x) = 1$ each time, return TRUE.

This algorithm is complete, since for the function $g(x) = 1$, we accept g always. For soundness, we assume our function h is ϵ-far from $\mathcal{P}$ so we have that $\Pr_x(h(x) = 1) < 1 - \epsilon$. Then the probability of falsely reporting TRUE is at most $(1-\epsilon)^{2/\epsilon}$. Observe that in the interval $[0, 1]$, this is maximal at 0, where it is a fixed constant less than $\frac{1}{3}$. □

10.4.1 Linearity Testing

We will now analyze a more complex property called linearity. According to O'Donnell, probabilistically checkable proofs (PCP's) are an offshoot of linearity testing.

Theorem 10.14. *(Linearity Testing) $\mathcal{P} = \{f : \forall x, y,\, f(x)f(y) = f(x \cdot y)\}$ is ϵ-testable in $O\left(\frac{1}{\epsilon}\right)$ queries, where $x \cdot y$ denotes element-wise multiplication.*

This problem is known as linearity testing, since we are testing for linearity of f under the multiplication operation. It is also helpful to note that more classically, if we used $\{0, 1\}$ with mod-2 addition instead of $\{-1, 1\}$ with multiplication, then the equivalent condition is if $f(x) + f(y) \equiv f(x + y) \pmod 2$ for $x, y \in \mathsf{GF}^n(2)$.

We will first establish relate linearity testing to our earlier analysis of parity functions:

Lemma 10.15. *$\mathcal{P}$ is precisely the set of all parity functions.*

Proof. For a parity function $f = \mathcal{X}_S$, $f(x \cdot y) = \prod_{i \in S} x_i y_i$ while $f(x)f(y) = \prod_{i \in S} x_i \prod_{i \in S} y_i$. Then clearly $f(x \cdot y) = f(x)f(y)$.

For the other direction, we assume that our function f satisfies $f(x \cdot y) = f(x)f(y)$. Let $e^{(i)}$ be the vector such that $e^{(i)}_i = -1$ and $e^{(i)}_j = 1$ at all $j \neq i$. Then each $f(e^{(i)}) \in \{-1, 1\}$; we claim that $f = \mathcal{X}_S$ for

$S = \{i : f(e^{(i)}) = -1\}$. But for any vector x, we can take the coordinates $\mathcal{I} = \{j : x_j = -1\}$, using which we can decompose

$$x = \prod_{j \in \mathcal{I}} e^{(j)}$$

where the product is taken using element-wise multiplication. Then by assumption,

$$f(x) = \prod_{j \in \mathcal{I}} f(e^{(j)}) = \prod_{j \in \mathcal{I} \cap S} (-1) = (-1)^{|\mathcal{I} \cap S|}$$

Thus $f(x)$ is -1 iff x has an odd number of -1 within the variables of S, so by definition $f = \mathcal{X}_S$. □

We will now proceed to prove the main theorem.

Proof. Our algorithm, which reminds of the one we saw above, is given as follows:

Repeat $\frac{2}{\epsilon}$ times:

1. Choose $x, y \in \{-1, 1\}^n$ independently and uniformly at random.
2. Query $f(x), f(y), f(x \cdot y)$
3. If $f(x \cdot y) \neq f(x)f(y)$, then return False.

After all queries, return True.

Completeness. This is trivial, since for every $f \in \mathcal{P}$, by definition $f(x \cdot y) = f(x)f(y)$ always, so the algorithm certainly returns True.

Soundness. Suppose f is a Boolean function that is ϵ-far from $\mathcal{P}$. Let Accept be the event that the algorithm returns true on f. Then we can write

$$\Pr(\textsc{Accept}) = \mathop{\mathbb{E}}_{x,y} \left[\frac{1}{2} + \frac{1}{2} f(x)f(y)f(x \cdot y) \right]$$

Since $\frac{1}{2} + \frac{1}{2} f(x)f(y)f(x \cdot y)$ is an indicator for $f(x)f(y) = f(x \cdot y)$. Then, taking the Fourier transform of f:

$$\begin{aligned}
\Pr(\textsc{Accept}) &= \frac{1}{2} + \frac{1}{2} \mathop{\mathbb{E}}_{x,y} \left[\left(\sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i \right) \left(\sum_{T \subseteq [n]} \hat{f}(T) \prod_{i \in T} y_i \right) \left(\sum_{U \subseteq [n]} \hat{f}(U) \prod_{i \in U} x_i y_i \right) \right] \\
&= \frac{1}{2} + \frac{1}{2} \sum_{S,T,U \subseteq [n]} \hat{f}(S)\hat{f}(T)\hat{f}(U) \mathop{\mathbb{E}}_{x,y} \left[\prod_{i \in S} x_i \prod_{i \in T} y_i \prod_{i \in U} x_i y_i \right] \\
&= \frac{1}{2} + \frac{1}{2} \sum_{S,T,U \subseteq [n]} \hat{f}(S)\hat{f}(T)\hat{f}(U) \mathop{\mathbb{E}}_{x,y} \left[\left(\prod_{i \in S} x_i \prod_{i \in U} x_i \right) \left(\prod_{i \in T} y_i \prod_{i \in U} y_i \right) \right]
\end{aligned}$$

Recalling the symmetric difference trick from when we proved orthogonality of the parity functions. Since $x_i^2 = y_i^2 = 1$ always and using $A \Delta B$ denotes the symmetric difference between two sets, i.e. $(A \cap \overline{B}) \cup (\overline{A} \cap B)$,

$$\begin{aligned}
\Pr(\textsc{Accept}) &= \frac{1}{2} + \frac{1}{2} \sum_{S,T,U \subseteq [n]} \hat{f}(S)\hat{f}(T)\hat{f}(U) \mathop{\mathbb{E}}_{x,y} \left[\prod_{i \in S \Delta U} x_i \prod_{i \in T \Delta U} y_i \right] \\
&= \frac{1}{2} + \frac{1}{2} \sum_{S,T,U \subseteq [n]} \hat{f}(S)\hat{f}(T)\hat{f}(U) \prod_{i \in S \Delta U} \mathbb{E} x_i \prod_{i \in T \Delta U} \mathbb{E} y_i && \text{Independence} \\
&= \frac{1}{2} + \frac{1}{2} \sum_{S,T,U \subseteq [n]} \underbrace{\hat{f}(S)\hat{f}(T)\hat{f}(U) \prod_{i \in S \Delta U} 0 \prod_{i \in T \Delta U} 0}_{0 \text{ unless S = U = T}} && x_i, y_i \text{ chosen u.a.r.}
\end{aligned}$$

Observe that the products are zero unless empty, in which case they are one. Thus the entire quantity in the summation is zero except for when $S = U = T$. Then we can simplify the above expression dramatically by getting rid of T, U and making the summation only in terms of S:

$$\begin{aligned}
\Pr(\textsc{Accept}) &= \frac{1}{2} + \frac{1}{2} \sum_{S \subseteq [n]} \hat{f}(S)^3 \\
&\leq \frac{1}{2} + \frac{1}{2} \max_S \{\hat{f}(S)\} \sum_{S \subseteq [n]} \hat{f}(S)^2
\end{aligned}$$

Using Parseval's theorem (theorem 10.9),

$$\Pr(\textsc{Accept}) \leq \frac{1}{2} + \frac{1}{2} \max_S \{\langle f, \mathcal{X}_S \rangle\} \tag{10.4}$$

Recalling that we defined $\langle f, \mathcal{X}_S \rangle = \mathbb{E}_x[f(x)\mathcal{X}_S(x)]$, which is equal to

$$\begin{aligned}
\langle f, \mathcal{X}_S \rangle &= \mathbb{E}_x[f(x)\mathcal{X}_S(x)] \\
&= \underbrace{\mathbb{E}_x[f(x)\mathcal{X}_S(x) | f(x) = \mathcal{X}_S(x)]}_{1} \Big(\Pr\big(f(x) = \mathcal{X}_S(x)\big) \Big) + \\
&\quad \underbrace{\mathbb{E}_x[f(x)\mathcal{X}_S(x) | f(x) \neq \mathcal{X}_S(x)]}_{-1} \Big(1 - \Pr\big(f(x) = \mathcal{X}_S(x)\big) \Big) \\
&= 2\Pr\big(f(x) = \mathcal{X}_S(x)\big) - 1
\end{aligned}$$

But by assumption, f is ϵ-far from $\mathcal{P}$, so $\Pr(f(x) = \mathcal{X}_S(x)) \leq 1 - \epsilon$, meaning

$$\langle f, \mathcal{X}_S \rangle \leq 2(1 - \epsilon) - 1 = 1 - 2\epsilon$$

Note that this is true for any S, so we can use it to replace the maximum in (10.4):

$$\Pr(\textsc{Accept}) \leq \frac{1}{2} + \frac{1}{2}(1 - 2\epsilon) = 1 - \epsilon$$

This is the probability of false positive in a single iteration. The algorithm consists of $\frac{2}{\epsilon}$ iterations, so the probability of failure is reduced to at most $(1 - \epsilon)^{2/\epsilon}$, which as noted before is below $\frac{1}{3}$ as required by the testable condition. Also note that we will make a total of $3\frac{2}{\epsilon} = O\left(\frac{1}{\epsilon}\right)$ queries. □

10.5 Application: Arrow's Impossibility Theorem

As a fun application, we will next use Boolean Fourier analysis for political theory, specifically in the analysis of elections. For simplicity we will only consider 3-candidate Condorcet elections.

For the 3-candidate Condorcet election among n voters, we assume each citizen i has a *strict* total order $<_i$ of the 3 available candidates, for example $A <_i B <_i C$, in mind. A *strict total order* is like a total order, as we have seen in a previous lecture, but the binary relation is strict, and we require that for elements $X \neq Y$, always exactly one of $X < Y$ or $Y < X$ holds. Practically, this means we can sort the candidates list without any ties. For each pair of candidates $(X, Y) \in \{(A, B), (B, C), (C, A)\}$, we collect the voter' preferences into a vector $v^{(X,Y)} \in \{-1, 1\}^n$, where if citizen i has $A <_i B$, $v_i^{(X,Y)} = -1$, and else $v_i^{(X,Y)} = 1$.

We consider the social objective of election to be building a new strict total order $<$ of all the candidates from all the citizen's choices. To be as general as possible, we model the election process by a Boolean *choice function* $f : \{-1, 1\}^n \to \{-1, 1\}$ which takes all the citizens' preferences into account. We set $X < Y$ if $f(v^{(X,Y)}) = -1$, else set $Y < X$ if $f(v^{(X,Y)}) = 1$.

Some examples of choice functions are:

1. $f(x) = 1$. Always pick the first candidate over the second, ignoring all voter preferences. (This always results in contradictions, so isn't really valid)
2. $f(x) = \textsc{Maj}(x)$. Take the majority vote.
3. $f(x) = x_i$. We call this a *dictatorship*. In other words, a special citizen i always decides the outcome of the election, and everyone else has their preference ignored.

We desire the election to have several seemingly innocuous properties:

Definition 10.16. *(Rationality) A choice function f is rational if regardless of citizens' preferences, the resulting ordering $<$ of candidates is always a strict total order. For example, we cannot have $A < B < C < A$.*

In other words, the results of the election should be consistent and not produce contradictions.

Definition 10.17. *(Unanimity) f satisfies unanimity if $f(1, 1, \ldots, 1) = 1$ and $f(-1, -1, \ldots, -1) = -1$*

In other words, if every citizen prefers a candidate over another, the latter should not be elected over the former.

Perhaps surprisingly, these two conditions cannot be simultaneously satisfied unless our election system is a dictatorship, as seen in the theorem below:

Theorem 10.18. *(Arrow's Impossibility) Let the Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$ be the choice function for a 3-candidate Condorcet election. If f is rational and satisfies unanimity, then f is a dictatorship.*

Note: while it is possible to prove this theorem using other methods, Boolean Fourier analysis provides a clean approach.

For convenience, we define Fourier weight $\mathcal{W}_k$ as follows:

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

Proof. Let $R(a,b,c)$ be the indicator $R : \{-1,1\}^3 \to \{-1,1\}$ which is equal to -1 iff (a,b,c) is a *rational* preferences, that is, results in a valid strict total order. Since there are only $2^3 = 8$ choices of (a,b,c), let's just brute force them all.

We find that the rational preferences for 3 candidates are exactly:

$$(1,1,-1),\ (1,-1,1),\ (1,-1,-1),\ (-1,1,1),\ (-1,1,-1),\ (-1,-1,1)$$

The two irrational preferences are simply $(1,1,1), (-1,-1,-1)$.

Let us take the Fourier transform of R:

$$\begin{aligned}
R(a,b,c) &= 1 \cdot \left(\frac{1}{2}+\frac{a}{2}\right)\left(\frac{1}{2}+\frac{b}{2}\right)\left(\frac{1}{2}+\frac{c}{2}\right) \\
&+ (-1) \cdot \left(\frac{1}{2}-\frac{a}{2}\right)\left(\frac{1}{2}+\frac{b}{2}\right)\left(\frac{1}{2}+\frac{c}{2}\right) \\
&+ (-1) \cdot \left(\frac{1}{2}+\frac{a}{2}\right)\left(\frac{1}{2}-\frac{b}{2}\right)\left(\frac{1}{2}+\frac{c}{2}\right) \\
&+ (-1) \cdot \left(\frac{1}{2}+\frac{a}{2}\right)\left(\frac{1}{2}+\frac{b}{2}\right)\left(\frac{1}{2}-\frac{c}{2}\right) \\
&+ (-1) \cdot \left(\frac{1}{2}+\frac{a}{2}\right)\left(\frac{1}{2}-\frac{b}{2}\right)\left(\frac{1}{2}-\frac{c}{2}\right) \\
&+ (-1) \cdot \left(\frac{1}{2}-\frac{a}{2}\right)\left(\frac{1}{2}+\frac{b}{2}\right)\left(\frac{1}{2}-\frac{c}{2}\right) \\
&+ (-1) \cdot \left(\frac{1}{2}-\frac{a}{2}\right)\left(\frac{1}{2}-\frac{b}{2}\right)\left(\frac{1}{2}+\frac{c}{2}\right) \\
&+ 1 \cdot \left(\frac{1}{2}-\frac{a}{2}\right)\left(\frac{1}{2}-\frac{b}{2}\right)\left(\frac{1}{2}-\frac{c}{2}\right) \\
&= \frac{1}{2}(ab+ac+bc-1)
\end{aligned}$$

Now let us consider a random election, where the voters' choices $v^{(A,B)}, v^{(B,C)}, v^{(C,A)}$ are such that each $(v_i^{(A,B)}, v_i^{(B,C)}, v_i^{(C,A)})$ is i.i.d. drawn u.a.r. from all rational preferences (since we assume each voter's preferences form a strict total ordering; note that the voters also only have the above 6 possible choices). For conciseness we will use x, y, z to denote $v^{(A,B)}, v^{(B,C)}, v^{(C,A)}$, respectively. By the rationality assumption, the election outcome must also be always rational, so

$$\begin{aligned}
-1 &= \mathbb{E}[R(f(x), f(y), f(z))] \\
&= \frac{1}{2}\mathbb{E}[f(x)f(y) + f(x)f(z) + f(y)f(z) - 1] \\
2 &= -3\mathbb{E}[f(x)f(y)] + 1 && \text{Symmetry} \\
\mathbb{E}[f(x)f(y)] &= -\frac{1}{3}
\end{aligned}$$

Next, we will take the Fourier transform again, this time for f:

$$\begin{aligned}
\mathbb{E}[f(x)f(y)] &= \sum_{S,T\subseteq[n]} \hat{f}(S)\hat{f}(T)\mathbb{E}[\mathcal{X}_S(x)\mathcal{X}_T(y)] \\
&= \sum_{S\subseteq[n]} \hat{f}(S)^2\mathbb{E}[\mathcal{X}_S(x)\mathcal{X}_S(y)] && \text{Theorem 10.7} \\
&= \sum_{S\subseteq[n]} \hat{f}(S)^2 \prod_{i\in S} \mathbb{E}[x_i y_i] && \text{Independence} \\
&= \sum_{S\subseteq[n]} \hat{f}(S)^2 \prod_{i\in S} (\Pr(x_i = y_i) - \Pr(x_i \neq y_i)) \\
&= \sum_{S\subseteq[n]} \hat{f}(S)^2 \prod_{i\in S} \left(\frac{1}{3} - \frac{2}{3}\right) && \text{Inspection} \\
&= \sum_{S\subseteq[n]} \left(-\frac{1}{3}\right)^{|S|} \hat{f}(S)^2 \\
&= \sum_{k=0}^{n} \left(-\frac{1}{3}\right)^k \mathcal{W}_k(f) && \mathcal{W}_k(f) = \sum_{|S|=k} \hat{f}(S)^2
\end{aligned}$$

Thus

$$\sum_{k=0}^{n} \left(-\frac{1}{3}\right)^k \mathcal{W}_k(f) = -\frac{1}{3} \tag{10.5}$$

But by Parseval's theorem (10.9),

$$\sum_{k=0}^{n} \mathcal{W}_k(f) = \sum_{S\subseteq[n]} \hat{f}(S)^2 = \langle f, f\rangle = 1 \tag{10.6}$$

Noting that for any x, $f(x) \in \{-1, 1\}$ so $f(x)^2 = 1$. Thus in (10.5), $\mathcal{W}_k(f)$ are a convex combination of $\left(-\frac{1}{3}\right)^k$ and equal $-\frac{1}{3}$, meaning it must be that $\mathcal{W}_1(f) = 1$ and all other $\mathcal{W}_k(f) = 0$, so the only nonzero $\hat{f}(S)$ are where $|S| = 1$.

Next, assume for sake of contradiction that $0 < \hat{f}(\{i\})^2 < 1$ for some i. Then we have

$$1 = \sum_{i=1}^{n} \hat{f}(\{i\})^2 < \sum_{i=1}^{n} |\hat{f}(\{i\})| = \sum_{i=1}^{n} f(\{i\})\text{sign}(\hat{f}(\{i\})) = f(x^*)$$

if we choose $x_i^* = \text{sign}(\hat{f}(\{i\}))$. But this means $f(x^*) \notin \{-1, 1\}$, contradiction.

Thus all $\hat{f}(\{i\})$ are 0 or 1. Since they sum to 1, exactly one $\hat{f}(\{i_0\}) = \pm 1$ while every other $\hat{f}(\{i\}) = 0$. Then the Fourier expansion of $f(x)$ can be simplified to:

$$f(x) = \hat{f}(\{i_0\})x_{i_0}$$

By assumption of uniformity, $f(1) = 1$, so it must be that $\hat{f}(\{i_0\}) = 1$, or in other words $f(x) = x_{i_0}$. Thus f is a dictatorship. □